THE USE OF RATINGS FOR THE INTEGRATION OF PLANNING & LEARNING IN A BROAD BUT SHALLOW AGENT ARCHITECTURE

Christian James Paterson[1]

A thesis submitted to the Faculty of Science
of The University of Birmingham
for the degree of
MASTER OF PHILOSOPHY

Cognition and Affect Group[2]
Department of Cognitive Science
Faculty of Science
The University of Birmingham
February, 1995

---

1 Funded by a Birmingham University Computer Science Department Studentship.

2 The Cognition and Affect Group includes: Prof. A. Sloman, Prof. G. Humphreys, E. Shing, T. Read, I. Wright and C. Complin. It is supported by the UK JCI (grant ref. MRC SPG9200393) and the Renaissance Trust.

ABSTRACT

The effective integration of both planning and learning must be viewed as a prerequisite to the creation of a truly intelligent autonomous agent, one of AI's Holy Grails. Many approaches, implemented within many systems, have been propounded yet all have fallen short of the mark. The proposed AIMAE[3] system, a broad but shallow agent architecture, provides just such an integration in a situated, goal-directed fashion. This is made possible via the use of behaviour-based ratings providing a multi-dimensional ordering on sub-plans, and hence acting as heuristic guides to plan construction. Furthermore, use is made of a control plan mechanism which, it is hoped will allow the system to address multiple concurrent goals, and carry out a degree of opportunity taking. In the course of investigating the constraints and requirements that apply to intelligent autonomous behaviour, especially that involving planning and learning, the following paper examines a number of landmark systems before going on to propose the above mentioned AIMAE system. Finally, a look is taken at how such a system can aid in the redefining and exploration of a variety of psychological terms.

AIMAE is an architecture that, I believe, will extend the boundaries to which other systems have been limited. Hence, redressing the general scientific trend of fragmenting research, in the hope that light may be shed on the psychology of intelligent animals such as ourselves.

3 AIMAE = Agent Information Management And Execution.

WITH THANKS TO

I would like to thank A. Sloman and I. Wright for help in preparation of this paper.

CONTENTS

1. Introduction 1
1.1. Research Motivation 1
1.2. Command And Control 2
1.3. The Design Based Approach 3
1.4. This Paper 6
2. In Search Of A Possible Design 8
2.1. Environmental Constraints 8
2.2. Physical Constraints 15
2.3. Cognitive Constraints 16
2.4. Summary 18
3. Requirements For Autonomy 20
3.1. General Requirements 20
3.2. Information Gathering Requirements 21
3.3. Goal Filtering Requirements 21
3.4. Goal Assessment Requirements 22
3.5. Goal Scheduling Requirements 23
3.6. Goal Processing Requirements 23
3.7. Summary 24
4. Traditional AI 26
4.1. The Symbol System Hypothesis 26
4.2. Classical Planning 30
4.3. Problems With Classical Planning 33
4.4. Reactive Planning 35
4.5. Summary 36
5. Integrated Architectures 38
5.1. Model-Based Reasoning 39
5.2. The BDI Formalism 41

5.3. NML1 44
5.4. Dynamic Control Reasoning 47
5.5. Summary 49
6. NOUVELLE AI 51
6.1. Issues Concerning Representation 51
6.2. Problems With The Physical Grounding Hypothesis 53
6.3. Issues Concerning Agent Structure 56
6.4. Problems The Behaviour-Based Approach 58
6.5. Summary 61
7. The AIMAE Architecture 63
7.1. The Architecture 64
7.2. Goal Management Module 66
7.3. Various Data Stores 94
7.4. Summary 95
8. Possible Scenarios 97
8.1. The Nursery Scenario 97
8.2. Other Possible Scenarios 100
9. Conclusion 101
9.1. Real-World Command & Control 101
9.2. The AIMAE System 103
9.3. Insights Into Psychology 106
10. Appendices A - E 110
11. References 179

DIAGRAMS

Fig. 4.1	A Symbolic Representation	26
Fig. 6.1	The Possible Control Network For A Simple Limb	57
Fig. 7.1	The Mapping Between Virtual & Physical Architectures	65
Fig. 7.2	Scematic View Of The Internal Divisions Within Aimae	65
Fig. F.3	Hierarchical Control Within AIMAE	68
Fig. 7.4	Goal Flow	69
Fig. 7.5	Control Plan Structure	71
Fig. 7.6	A Plan Tree	72
Fig. 7.7	The Strategies For The Plan In Fig. 7.6	74
Fig. 7.8	A Strategy	76
Fig. 7.9	Cross Indexing Look-Up Tables	77
Fig. 7.10	Multiple Strategies May Solve The Same Goal	78
Fig. 7.11	Goals, Strategies & Ratings	79
Fig. 7.12	The Use Of Ratings In Strategy Selection	82
Fig. 8.1	A Nine Room Nursery	99
Fig. A.1	From Perception To Action (And Back)	110
Fig. B.1	Categorizing Goals	117
Fig. E.1	Pengi's Internals	140
Fig. E.2	Decomposition Of A Sequential Circuit	149
Fig. F.1	Goal Generation & Activation Levels	164
Fig. F.2	From Domain Events To System Goals	165

1. INTRODUCTION

1.1 Research Motivation

From the machiavellian Hal in 2001: A Space Odyssey, to the harmless C3-PO in Star Wars, the spectre of intelligent machines has stalked the pages of science fiction since the genre was born. Indeed, the idea that just around the corner intelligent machines will be capable of doing everything humans can, and more, has enticed both children and adults with abandon into imaginary future worlds; worlds where humans and machines are inseparable. Such 'thinking' machines are viewed as our liberators, helping to free mankind from both the drudgery of simplistic and mundane tasks, as well as the limitations imposed by our own physical, conceptual and computational capabilities.

In addition to these 'romantic' roles, in much the same way that the development of atomic theory gave scientists a means of better understanding the nature of matter, it is entirely plausible that investigation into the possible designs for autonomous intelligent agents, will reveal far reaching insights into our own cognitive apparatus (Sloman & Croucher, 1981; Sloman 1991a, 1991b, 1991c, 1992b, 1994a). Insights that will aid in the provision of a less abstract foundation from which behavioural theories can be developed (e.g. Correa & Coelho 1992). Hence, such investigation will facilitate a greater understanding of how and why cognitive mechanisms elicit maladaptive and adaptive behaviour. Indeed, it will allow the disambiguation of terms used to describe many behavioural states and phenomena, e.g. emotions, consciousness and attention.

For these reasons, the task of designing an agent capable of autonomous, intelligent behaviour within a real world environment (or a facsimile of it), has been one of AI's holy grail's. Yet even with the advent of more powerful computers, few researchers have aimed at integrating a wide range of behavioural capabilities within agents inhabiting real-world environments. Most notably, while learning mechanisms proliferate within complex systems embedded within simple domains, or simple systems embedded within complex domains, the integration of learning mechanisms into complex real-world systems appears to have been neglected somewhat. This paper examines the techniques and landmark systems that have previously been put forward in attempting to achieve complex real-world systems, prior to putting forward the AIMAE system (Chapter 7). This is a system designed to integrate some of the behavioural capabilities that we might ascribe to humans within a 'broad but shallow' (Bates & Reilly 1991; Loyall & Bates 1991) real-world embedded agent. Most notably the AIMAE system suggests ideas for the integration of behaviour-based learning and goal directed planning.

1.2 Command And Control

There has existed for many years a debate not only within the animal behaviour and psychology literature, but also within more philosophical writings, concerning the manner in which an agent is able to transcribe perception into effector output, i.e. produce behaviour (see Appendix A). By delimiting the manner in which an agents command and control system operates, the approach opted for can be seen to have a bearing not only upon an agent's possible behaviours, but also upon their optimality, and thus the agent's overall viability. A good example of this would be the comparison between using a desktop publishing system and a text processor. The text processor being uncomplicated and sparse in its features is quick to learn, and quick to use. However, its simplicity is also its downfall - it cannot deal with complicated text layouts, diagrams, or pictures. Indeed, it is little more than a typewriter. By comparison, the desktop publishing system is far more adept at managing text flows, and pictures. Yet, just as the text processor's simplicity was its undoing, so to is the desktop publisher's complexity. Being complicated it is both difficult to learn and use. Thus, while a desktop publishing system is good for producing journal articles, it is unwieldy for writing a quick letter to a friend! In a similar vein we see the distinction between the different approaches used in the design of autonomous agent command and control systems.

The above highlights the fact that the problems facing an autonomous agent's command and control system requires a decision concerning what approach to take that is neither clear cut, or easy, but one where the benefits as well as the costs must be weighed up. These problems can be viewed as falling into four categories:

i. Task selection.

What task(s) do I attempt to achieve next?

ii. Action selection.

What potentially correct actions are available to me given my answer to (i) and the current circumstances?

iii. Action arbitration.

How do I resolve any conflicts between the actions selected in (ii)?

iv. Action execution.

How do I go about executing actions resulting from (iii) in a co-ordinated fashion?

Examination of the above four points reveals that the task of achieving command and control within an autonomous agent embedded within a real-world environment can be viewed largely as a planning problem (Chapman 1989). However, when deciding upon the mechanisms for such a control system, in addition to considering the above, the question of how such a system would be implemented must also be contemplated, i.e. what architectural form will the design take. Maes writes that an architecture...

> "proposes a particular methodology for building an autonomous agent. It specifies how the problem can be decomposed into sub-problems".
>
> *Maes, 1991, p. 115*

1.3 The Design Based Approach

The position taken with regard to systemic implementation will clearly influence, and in turn be influenced by, the stance taken in addressing the above points. For this reason it is likely that cognitive investigation will be best served via the utilisation of a design-based approach (e.g. Sloman 1993a), an approach which requires consideration of the following:

i. What architectural 'components' are necessary to produce the desired behavioural capabilities?

ii. How are such 'components' to be cohesively implemented within an agent?

For example, will the mechanisms for command and control be fully centralised (e.g. the majority of robot control systems), fully distributed (e.g. the Behaviour-Based AI systems), or a mixture of the two (e.g., the neuronal system of most chordate animals, for example humans)? Will action selection and arbitration schemes be fixed or dynamic? Will learning take place, and to what degree? Etc. Many questions must be addressed prior to, and concurrent with, considerations concerning these two questions.

While the term 'design-based approach' implies that investigation should be directed towards the physical designs of autonomous intelligent agents, this is not necessarily the case. Take as an analogy a word processor. Within it, words, sentences, paragraphs, and pages, are all configured to form a specific, but changeable architecture - the document. However, this architecture is said to be virtual, implemented by the word processor package rather than being physically present within the computer. In much the same way, the word processor itself is a virtual construction of windows, icons, and menus, based upon a lower level architecture made up of program procedures. Furthermore, this grounding of multiple levels of virtuality on top of each other, from the very abstract down to the concrete computer hardware, means that an inspection of the hardware is unlikely to reveal much about how and why certain user operations result in certain document changes. To gain an understanding of why the word processor behaves as it does one needs look not only at its manifest behaviour, and its hardware architecture, but also at the virtual levels separating the two.

For this reason, trying to gain knowledge concerning why an agent behaves as it does by looking solely at either its neurobiology (hardware), or its manifest behaviour, is going to be of restricted use in determining how and why such behaviour occurs; such studies are useful only in that they specify relatively clearly the two areas on the map that must be joined. From this we can see that for science to achieve an understanding of intelligent behaviour in anything but an abstract manner, development of theories concerning the virtual architectures that support such behaviour is required. Furthermore, these theories must be linked to the agent's underlying neural architecture.

Understandably, research has tended to fragment and simplify the problem gaining expertise in highly restricted domains at the expense of functionality in real-world environments, e.g.

planning, learning, perception, etc. However, architectures that lack the means of producing a wide range of behaviour may also lack the sophistication required by research for the formulation of correct, and/or complete answers to questions regarding internal mechanisms within biological agents (e.g. Pfeifer 1992). Take as an example, the contentious subject of agent perception and representation. To argue that perception always gives rise to explicit symbolic representation is as foolish as arguing the reverse. Surely, the forms that perception takes depends largely upon the needs of a specific agent, e.g. whether it requires fast reflexive action initiation or deliberative goal processing? Such uses cannot be fully known if the agent is not fully specified. In short, arguing over the form a limited number of internal mechanisms take when others have been left out leads research down the path of ambiguity and pointlessness. Yet, this is the very problem that has pervaded much of the research into agent functioning and cognition. The imposition of resource limitations (financial, technological, and temporal) upon research has tended to result in the detailed perlustration of agent sub-components (behavioural or architectural) at the expense of whole agent functionality. This is a trend which must be reversed.

The design-based approach adopted by the Cognition & Affect Group (Sloman & Humphreys 1992) attempts to redress this trend by investigating the variety of underlying command and control system architectures that supports the production and integration of behavioural capabilities within whole agents architectures. Following such a research methodology tends to give rise to architectures best described as 'broad but shallow' (Bates & Reilly 1991; Loyall & Bates 1991); intelligent autonomous agents who possess to a varying degree some of the abilities ascribed to humans, e.g. planning, learning, perception, and reaction.

However, if we are to fulfil the aim of clarifying 'psychological' and 'behavioural' theories to a point which is unachievable today we must fully understand the relationships between an agent's behaviour, its design, and its environment. For this to occur an examination of the variety of architectures that support such phenomena must be carried out, i.e. we must explore design space. Take for example the term 'attention'. The processes that attention may influence, or indeed be influenced by, are not yet well understood, and it is likely that such a broad and encompassing term is insufficient for the variety of phenomena that it is currently seen to account for. As such, the use of the term *attention* in anything but a naive and non-functional manner is fool-hardy, especially with respect to other abstract and ill-defined terms such as *consciousness* and *emotion*. To state that attention is *the* mechanism by which goals are selected for achievement, or parts of an image selected for processing, is the equivalent to stating that all Ferraris are red; it is unlikely in the extreme. It is far more plausible to believe that what we naively term attention is purely an emergent phenomenon arising out of the myriad of control systems that interact within 'intelligent' agents. Yet, it should be remembered that in the same way that words in a word processor do not physically exist and do not have any functional power, the phenomenon of attention existing as a virtual artifact may be quite plausible. What science needs to ascertain is not what attention (or emotion, or consciousness) is, but what processes and interactions give rise to it, and this is can only be achieved through an exploration of design space. The broad but shallow design approach gives rise to faster agent development, and hence allows for just such an exploration, giving us a stronger position from which to make accurate predictions and more clearly defined terminology.

1.4 This Paper

This paper is primarily concerned with the problems associated with the integration of situated activity and behaviour based learning within a real-world embedded broad but shallow agent. I put forward with the AIMAE system an autonomous agent's virtual command and control architecture for just such an integration. This is achieved through the use of behaviour-based ratings that order plan components along multiple dimensions for their use in plan construction. Furthermore, the proposed AIMAE system also extends the boundaries of other broad but shallow agents by supporting a notion of time as well as richer goal and plan manipulation mechanisms. These allow not only a dynamic balance between deliberation and reaction to be kept, but also support the pursuit of multiple conflicting goals and a degree of opportunity taking. The architecture is based primarily upon previous work done within the Cognition & Affect Group, most notably by A. Sloman and L.P. Beaudoin, as well as previous work carried out within the fields of planning, and agent architectures. Finally, this paper uses notions gleaned from a mixture of disciplines, especially those of Philosophy, Psychology, Neurobiology, and Computing. This paper takes the following structure:

i. The problem.

a. Chapter 2 - Constraints.

b. Chapter 3 - Requirements.

ii. Previous Solutions.

a. Chapter 4 - Traditional AI.

b. Chapter 5 - Integrated Architectures.

c. Chapter 6 - Nouvelle AI.

iii. My Solution.

a. Chapter 7 - The AIMAE Architecture.

b. Chapter 8 - Possible Scenarios.

iv. Chapter 9 - Conclusion.

2. IN SEARCH OF A POSSIBLE DESIGN

In an abstract sense, there exists a near infinite space of possible designs for objects that may exist within the world, bounded only by such physical laws as those governing time, space, and relativity. Indeed, there may exist possibilities that lie beyond our abilities to sense or conceptualise (cf. Kant). Such a space may be divided into subspaces of possible designs via the application of constraints, and furthermore, these subspaces may themselves be divided into subspaces, and so on. Hence, the division of the overall space tends to form a hierarchical structure, whose lower echelons define smaller subspaces than those higher up, e.g. the divisions utilised within taxonomy.

Therefore, the design and implementation of an agent's physical structure, as well as its command and control system, is determined by the size and complexity of the subspace within which that design is to be located. As we have seen, such a subspace is itself determined by the constraints placed upon the overall space of possible designs. The fewer and more general the required characteristics an agent must meet, the larger the space of possible agent architectures that will meet them. A such, increasing the number and specificity of the desired characteristics, leads to a reduction in this design space. A reduction which commensurably increasing the difficulty of 'finding' an agent that satisfies those desired characteristics. This is one of the reasons why so few agents have been designed to inhabit real world environments. Yet the environment is only one source of possible constraints, others may arise from the tasks we want our agent to achieve within that environment. For the purposes of this paper constraints are envisaged as falling into three broad categories: environmental, physical, and cognitive.

2.1 Environmental Constraints

In terms of the constraints placed upon an agent by its intended environment, the whole debate concerning whether an agent should be implemented within a real or simulated version of that environment is pointless so long as the two share the same essential characteristics, and place the same types of computational demands upon the agent. A real-world environment can be described as having two main defining characteristics, that is, it is both complex, and dynamic. Precise definitions of these two terms are hard to give, but will have many implications for the agent and its command and control system, e.g:

- Can several events co-occur?
- Can those events occur without the agent's knowledge?
- Is change within the environment continuous, for example the movement of the sun, or discrete, such as the moves in a game of chess?[4]
- Are there other agents and entities with the environment, for example humans (e.g. Brooks 1991a)?

It can be seen that these points are closely tied with both the degree that the environment operates asynchronously to the agent, and the level of abstraction that is used to observe it. The type of environmental complexity that I would like to use is one that implies that the environment is inhabited by other autonomous entities and not just the agent in question. The type of environmental dynamism will include, among other things, a notion of asynchrony, in that several events may co-occur, some without the agent's knowledge, and with continuous change. As such, I would not consider role playing games, for example chess, to be truly dynamic, and only marginally complex (for instance, the other agent working to a set of rules a subset of which are known). Of course describing domains with these terms can only be an imprecise and qualitative task, with the spectrum between the two ends of the scale being difficult to precisely quantify and describe. For example, some domains may be considered completely uncomplex, e.g. the Blocks World, while others may be highly complex, e.g. road traffic flow, but if so why and by how much is one more complex than the other? The opposite to this type of real-world domain (one which is both complex and dynamic, but not necessarily simulated or real) would be, for example the Blocks World scenario (e.g. Fahlman 1973, Sussman

---

4 Given that computers operate discretely, the exponents of computer simulation are forced into the assumption that continuous change can be modelled via discrete events. As such it can be argued that the importance of this point is negligible. However, the ease (or not) with which continuous events can be modelled does not mean that they can be effectively ignored, or treated in the same way that more discrete events are. A system should have the ability both to recognise the characteristics of events and be able to act accordingly. This is an integral part of intelligent planning; recognising, understanding, and utilising the dynamics of events to the agent's best advantage.

1975), which does not present an agent with the same types of physical and computational demands as would an agent within, for example, the Nursery scenario (Sloman 1991c & 1993b; Sloman & Humphreys 1992; Beaudoin & Sloman 1991 & 1993; Shing 1993). Thus, many of the previous studies that have been carried out into autonomous agents have been limited in terms of specifying the infrastructure required by an autonomous agent for it to survive viably within a real-world environmental niche (e.g. Brooks 1991a).

An agent embedded within a real-world environment will find itself constrained in how it reasons about the selection and execution of goals by a variety of environmentally specified parameters (Georgeff & Lansky 1987; Rao & Georgeff 1992). Although, it should be noted that these parameters become constraints only when the limits they impose are close to being reached. This aside, given that we wish to specify the design for an autonomous agent that should be capable of being placed within the real world (even if our final design is only implemented within a facsimile of that world) the following environmental parameters can be analysed to show how they will effect such a design.

2.1.1 Information Constitution

a. Data glut.

The quantity of information contained within the environment will mean that an agent is unlikely to be able to process all potentially interesting environmental events (Fehling et al. 1989; Hayes-Roth 1990b). This may be further exacerbated via the addition of more than one form of perceptual exteroceptive input, i.e. the agent may possess a number of environmental sensing devices, which may overlap in the physical quantities that they sense (Brooks 1986). A good example of this is the overlap between determining the orientation of an object both via touch and via vision.

Added to this data glut from the external environment of the agent, any agent which is to remain viable without on-going human intervention will need to be able to sense to some degree its own internal 'bodily' environment. For example, body temperature at global and local levels (e.g. Sloman & Croucher

19xx). Furthermore, an intelligent autonomous agent may also need to monitor its own cognitive activities so that it doesn't get into states such as 'thrashing' between goals. This form of self monitoring may require as much processing as (if not more than) environmental monitoring, and may potentially add large additional burdens to what is possibly an already over burdened control system.

It must be noted however, that such a data glut is not determined solely by the environment within which the agent finds itself. Equally important are the computational abilities that agent possess for dealing with incoming information. Given a powerful enough processor and the correct architecture, there may in fact exist no limit to the quantity of information that the agent could process. This is thus a practical consideration rather than a theoretical one, a point often overlooked.

b. Data distribution.

Data of potential interest to an agent within the environment may well be distributed over a wide area, and/or be mixed with a variable proportion of information which is irrelevant. Hence, an autonomous agent will require methods that allow it to search for, and extract data of potential interest from the environment. This problem is made worse by the fact that the data of potential interest to the agent at any moment in time will change depending on the current goals of that agent. Thus, in general, non-specific data retrieval strategies will always be non-optimal for an autonomous agent within a real-world environment. It should be noted that the use of active visual processes partially reduces this problem from one which is exponential to one which is linear (Tsotos 1989 - from Chapman 1989). Accordingly, the formation and use of implicit rather than explicit representations within the Behaviour-Based AI systems is one possible solution to this problem, although this in itself presents problems (see Section 6.2).

c. Events within the internal and external environment of an agent may map on to 'cognitive' events or states in diverse ways (Sloman 1994b). There may be many types of external events which trigger one type of cognitive event, or conversely one type of external event may trigger many different types of cognitive events (Hayes-Roth 1990b), and this would depend both upon the state of the agent and the state of the environment. For example, both an angry dog and a quickly approaching bus can elicit cognitive events which move the agent out of harm's way. Conversely, the quickly approaching bus may trigger a completely different goal, such as 'flag it down' if the agent wished to board it. Hence, the response an agent will make to environmental stimuli is likely to be neither stereotypical nor constant.

d. Data quality.

All devices designed to sense changes within a physical environment, regardless of whether they are artificial or natural, will suffer from a variable degree of inaccuracy (e.g. Brooks 1986; Gat 1991; Basye 1991), as well as resolution limitations. The manner in which raw sensory data is processed to produce usable perceptual input will effect the agent's perception of its environment, and the manner in which the agent then acts.

2.1.2 Dynamism

a. Multiplicity of conditions.

Within a real-world environment it would be impossible to enumerate all the possible situations and conditions that an agent could encounter (e.g. Ginsberg 1989). Indeed, this is one of the main failing points of Universal Planning (see Appendix C Section 1).

b. Real-time constraints.

Within a real-world environment events and entities will have their own temporal dynamics, and these will impose both hard and soft real-time constraint upon any agent that intends to act within that environment (e.g. Miller et al. 1985; Lesser et al. 1988; Fehling et al. 1989; Hayes-Roth 1990b; Morignot

1994; Lyons & Hendriks 1991). The crossing of the time boundaries imposed by these constraints will effect not only the utility of achieving specific goals, but also the overall optimality of the agent's behaviour (e.g. Pryor & Collins 1991). Soft real-time constraints can be defined as a point in time beyond which the utility (or benefit) of achieving a goal diminishes, whereas a hard real-time constraint could be defined as a point in time beyond which the goal is completely unachievable[5]. For example, if shops close at 6 PM and you wish to do a lot of shopping, then the utility of getting to them beyond (say) 5.30 PM is reduced. However, once 6 PM is reached the utility of getting to them is nil; the goal cannot be achieved. Hence, while the positioning of soft real-time constraints is in many cases determined by an agent's goals, the positioning of hard real-time constraints is determined by factors outside of an agent's control. Furthermore, the utility associated with crossing either boundary is determined by the value an agent places upon achieving the goal(s) in question. We can see therefore, that an autonomous agent should be capable of:

i. Assessing whether the achievement of a goal is theoretically possible given the imposed time constraints.

ii. Assessing whether the utility of attempting to achieve a theoretically possible goal outweighs any possible resource costs. The classical cost vs. benefit analysis.

iii. Behaving in a manner such that the achievement of goals meets imposed time constraints.

Of vital importance, therefore, to an autonomous agent is the necessity for it to behave in a *timely* manner. To quote Hayes-Roth (1990b, p.99), "behaviour is satisfactory not simply because it produces the correct result, but because it does so at the right time".

c. Potential interactions.

---

5 The term *Urgency* is often used to describe the effects that real-time constraints have upon the utility of behaviour (e.g. Beaudoin 1994; Hanks & Firby 1990; Sloman 1987).

i. Given that a complex simulated, or real, environment will contain other 'living' entities, it can be assumed that any agent that is to be placed within such an environment will have direct and indirect interactions with these entities. As mentioned above, other entities will have their own temporal dynamics that will place both hard and soft real-time constraints upon an agent. In addition to these time constraints, other entities may place resource constraints upon an agent such that it must compete for resources, or indeed protect previously attained goals against their premature undoing by such entities, e.g. a 'monster' in the Blocks World.

ii. The interaction of goals may occur with potentially disastrous effects. An agent should be aware of situations when the achievement of one goal is likely to lead to the undoing of a previously attained goal, or conversely the prevention of a later goal. Such potential interactions between goals should influence the manner in which an agent attempts to achieve its goals. For example, the temporal ordering that goals are given (e.g. Miller 1985; Dean et al. 1988; Bratman et al. 1988; Allen et al. 1983). One method which has recently been used with success within an autonomous agent is the use of Reference Tags by Pryor (1991).

Given both of the above types of possible interactions an agent should strive to maintain a global rather than a local co-ordination of actions, and this is one of the failings of both Reactive architectures and Behaviour-Based architectures (although work by Maes addresses this deficiency in Behaviour-Based systems - see Appendix E Section 4).

d. Diverse demands.

A complex and dynamic environment will always present a diversity of demands upon the agent in terms of how it is to interact and behave within such an environment, not least in terms of perception, interpretation, prediction, re-action, and planning (Hayes-Roth 1990b).

e. Variable stress.

The environment is likely to vary in its 'stressfulness' over time, i.e. the degree to which it present conflicting diverse demands to the agent (Beaudoin 1994).

f. Diversity of events.

Events and conditions within the environment will vary in their importance to an agent depending upon the agent's current beliefs, desire, and intentions (e.g. Perry 1985). However, it may be to the advantage of an agent to take note of information which is not immediately useful, not only for future reference, but also to prevent the agent from overlooking certain critical events and opportunities (e.g. Hayes-Roth 1990a & 1990b; Sloman & Croucher 19xx; Sloman 1987).

2.2 Physical Constraints

In addition to those constraints imposed upon an agent by its intended environment, the manner in which that agent is physically implemented may also impose constraints upon how that agent can execute actions (e.g. Brooks 1991c; Morignot 1994). These physical constraints will therefore influence the manner in which the agent selects and plans for the achievement of goals. For example:

- How many objects are there and how many 'hands' do you have?
- How wide is the gap, and how far can you jump?
- How hot is the drink, and at what temperature will your mouth burn?
- How much time do I have before the train leaves, and how fast can I run?

- How many things in different locations can I sense, and how many sensory inputs do I have, and how independent of each other and me are they?

2.3 Cognitive Constraints

An autonomous agent can be viewed as a computational system that has a set of goals and operates completely without outside intervention, e.g. from a human operator. As such, the behaviour of the agent will be greatly influenced by the manner in which the command and control system is designed and implemented. For example the agent's command and control system may face constraints in terms of:

i. Its current knowledge (e.g. Georgeff & Ingrand 1989; Fehling et al. 1989; Hanks & Firby 1990).

ii. Its ability to predict the future (e.g. Bratman et al. 1988; Morgenstern & Stein 1990; Shoham & McDermott 1988).

iii. Its ability to update any internal representations of the world that it utilises (e.g. Miller et al. 1985).

iv. Whether, and to what extent, processing is carried out in a serial or parallel fashion (e.g. Allen et al. 1983).

Each of these points is vitally important to the viability of an agent's command and control system, and as with the various routes to behaviour (see Appendix A), much as the above points determine this command and control system, so to the command and control system determines these points. Although, it would be useful to examine each of these points further, space does not permit such an exposition. However, within real-world environments a planning system's ability to fully and accurately predict future states influences not only the extent to which an agent is capable of planning for the achievement of its goals, but also the extent to which it may learn from any experiences it has while carrying out resultant plans. It is therefore clearly a point which cannot be overlooked. Perlustration reveals that predictability can be seen as being based upon three main factors[6]:

i. The complexity and dynamism of the environment (e.g. Agre & Chapman 1987; Firby 1987; Basye 1991; Shoham & McDermott 1988).

ii. The completeness of the system's domain knowledge (e.g. Dean et al. 1988; Horvitz 1989; Hanks & Firby 1990).

iii. The computational powers of the system (e.g. Fehling et al. 1989; Hanks & Firby 1990).

These points interact intimately. Allow me to clarify. Let us say for a moment that a hypothetical agent has complete knowledge of its domain, from the domain's state each and every moment, to information concerning how every entity within that domain may act and interact. Given this knowledge one might presume that the agent would be able to predict accurately the future state of the world given an event x. Yet this may not be the case. A prediction of this type would take time to produce, where the time required would be a function not only of the complexity and dynamism of the domain, but also the knowledge that the agent could utilise, and the computational power that the agent could bring to bear upon the problem. The more complex and dynamic the domain, the more events which may occur, and the greater the branching factor of the problem space through which search must work to produce a prediction; prediction being viewed as any other problem solving task. Furthermore, the agent's knowledge would also contribute to this branching factor. Hence, we can see that the predictability of a domain is *inversely proportional* to (an agent's knowledge concerning that domain, and) that domain's complexity and dynamism. Conversely, a domains predictability can be seen as being *proportional* to an agent's computational powers.

For the above reasons, all things being equal, the greater the complexity and dynamism of a domain, the lower an agent's ability to predict future events within that domain will be, and hence the lower the agent's ability to accurately produce a space of possible world models through which problem solving search must take place, i.e. the lower the agent's ability to accurately plan. Thus, when an agent's ability to probabilistically predict future events within the environment (including the outcome of its own actions) is removed, it removes the ability for an agent to do anything more than react to environmental events (Agre & Chapman 1987;

---

[6] Ignoring the additional problems posed by predicting chaotic systems!

Firby 1987; Lyons & Hendriks 1991), i.e. the agent effectively becomes a 'reactive planner' (see Section 4.4).

In terms of how domain predictability effects the utility associated with learning one has to consider the costs involved both in learning, and the retrieval of learnt knowledge, balanced against the applicability of that knowledge. Indeed, Minton (1985) has made a quantitative analysis of the costs involved in plan 'chunking' which showed that unless such chunking was constrained in its use, the costs involved in 'chunk' retrieval could grow to outweigh the benefits of not having to plan in the first place! It is clear that the predictability of a domain, and how that predictability then effects an agent's actions will determine one side of the cost vs. benefit analysis of learning.

2.4 Summary

In the previous sections an in-depth exposition of the plethora of constraints placed upon an agent embedded within a real-world environment was given, where those constraints were divided into three main categories: environmental, physical, and cognitive. What follows is a summary of those constraints:

i. Data glut.

Regardless of the computational abilities of an agent, it is unlikely to be capable of processing all the information perception can deliver.

ii. Data distribution.

Potentially interesting data may be distributed within an environment and map on to internal events in a variety of manners.

iii. Data quality.

Perceptual data is likely to suffer both from noise and various limitations.

iv. Multiplicity of Conditions.

Innumerable situations may arise within an environment.

v. Real-Time Constraints.

An agent will be constrained in the time it has for achieving goals.

vi. Potential Interactions.

An agent may be faced with interactions with other entities, or between the goals that it wishes to pursue.

vii. Variable stress.

An environment will vary in its stressfulness.

viii. Diversity of events.

Events and entities within an environment will vary with regards to their importance to an agent.

ix. Predictability.

The complexity and dynamism of an environment may result in it being unpredictable.

x. Physical Constraints.

An agent will be limited in what it physically can achieve in any one moment of time.

xi. Cognitive Constraints.

An agent will be limited in its cognitive capabilities.

The constraints summarised above determine to a large extent the behavioural capabilities that an agent embedded within a real-world environment must exhibit, capabilities that are extra to those which we, as system designers, might wish our system to have. These capabilities are made explicit within the next chapter's exposition of system requirements.

3. REQUIREMENTS FOR AUTONOMY

In the previous chapter it was shown that an autonomous agent embedded within a real-world environment will suffer a myriad of constraints upon the way it may behave; not least, environmental, physical, and cognitive. These constraints can be viewed as determining to a large extent the requirements that an agent's design should attempt to address, and the following makes these explicit.

3.1 General Requirements

i. The behaviour selection mechanisms that are employed within an agent's command and control system must ensure that the long term designer's goals for that agent are met.

ii. The behaviour of an agent should be 'salient' to the situation the it finds itself in.

iii. An agent is likely to suffer from a number of resource limitations. Consequently, it should act in accordance with the results of predictive reasoning, rather than acting 'blindly' and wasting constrained resources.

iv. An agent should have the ability to learn. An agent without the ability to learn from past experience will be limited in the manner with which it can deal with future situations, being forever locked in to behaving in a manner determined by the agent's designer. Such an agent is likely to be both non-optimal in terms of the cost vs. benefits of its actions, and in terms of its adaptability to unanticipated situations (cf. Universal Planning - see Appendix C Section 1).

v. Given that an agent is likely to suffer from data glut, along with real-time constraints, the rate at which internal events occur should be independent of the rate of events within the environment (Hayes-Roth 1990b). Furthermore, events within the various parts of the agent's command and control system should also operate asynchronously with respect to each other for the same reasons.

vi. Because of this internal asynchrony the various parts of the agent's command and control system should not only 'buffer' their communication channels, but have 'local' working memory sites, rather than operating on a single memory area, as with blackboard systems.

3.2 Information Gathering Requirements

i. To reduce any data glut that an agent may suffer, the perceptual mechanisms should be able to extract sensory data that is likely to be relevant to the agent given its current extant goals, e.g. have some form of active vision.

ii. To operate, an agent's command and control system does not require raw sensory data, but information concerning its environment. As such its perceptual mechanisms should carry out a degree of data processing prior to that data being passed to other mechanisms. A secondary benefit of such data interpretation is that the problems of data glut, signal noise, and inter-modal information overlap will be reduced.

iii. While carrying out the above (information selection and processing) the perceptual mechanisms should not remove, or reduce, the agent's sensitivity to either critical events, or information which may be useful at a later point in time (Hayes-Roth 1990). Critical events may be defined as those events which have a high utility with respect to the agent's survival, e.g. an angry person holding a baseball bat.

iv. The representational scheme that perceptual information is transcribed into should be determined by the agent's current goals, since the representational scheme has a major bearing upon the 'ease' and content of information that can be extracted. Hence, it is likely that more than one representational scheme will be used within an agent.

3.3 Goal Filtering Requirements

i. The perceptual and cognitive mechanisms are likely to deliver to other mechanisms information that will trigger a number of goals. Assuming various cognitive resource constraints, an agent should have some means of limiting the number of goals competing for those resources, i.e. an agent should have some form of pre-deliberative goal filtering.

ii. To be effective in reducing the load upon deliberative processes, goal filtering should be carried out without recourse to any such deliberation processes, i.e. goals need to be filtered heuristically (Sloman 1987).

iii. The filter's threshold should not be set at a constant value, rather it should be based upon the current computational loads that the system is supporting.

3.4 Goal Assessment Requirements

i. Prior to attempting to achieve them, the agent's command and control system should have the ability to reject post-filter goals on the basis of more 'informed considerations' concerning those goals, i.e. it should be able to decide which goals to *adopt* or *reject* (e.g. Beaudoin 1994).

ii. Adoption/rejection considerations should take into account whether the achievement of a goal is theoretically feasible given any imposed constraints, e.g. real-time constraints.

iii. Furthermore, goal consideration should include an assessment concerning whether the potential utility of attempting to achieve a theoretically possible goal outweighs the potential costs of doing so, i.e. goals should have some form of *importance* measure ascribed to them (Sloman & Humphreys 1992; Beaudoin 1994).

iv. The system should have the ability of carrying out such adoption considerations heuristically, or deliberatively depending on the goal in question, those goals already extant, and the time available, i.e. the *urgency* of the situation (Sloman & Humphreys 1992; Beaudoin 1994).

3.5 Goal Scheduling Requirements

i. In order for an agent to remain viable its adopted goals should be scheduled in some manner such that those of greater urgency and/or importance are made *active* prior to those of lesser urgency and/or importance.

ii. To allow an agent to effectively cope with both changing environmental circumstances and new goals, any schedule that an agent's command and control system utilises should be dynamically re-orderable .

iii. Since an agent is likely to suffer from a number of resource constraints, partially achieved, but suspended goals should not be discarded, but instead they should remain within the schedule for possible future re-activation.

iv. It is not enough that an agent acts in a goal directed manner in accordance with the current situation, it must also be capable of noting and utilising opportunities as and when they arise.

v. An agent should not rapidly switch between inconsistent behaviours, nor should two interfering behaviours be active concurrently, i.e. actions should be selected c'est selon (that is according to the circumstances).

3.6 Goal Processing Requirements

i. The command and control system must ensure that the agent's behaviour is directed towards its goals.

ii. Since behaviour should be both salient to the current situation in which the agent finds itself (requirement 3.1.ii), adaptive and optimal (requirement 3.1.iv), plans for active goals should be expanded on the basis of past experience and the current domain state.

iii. Reasoning about how to achieve (expand) a goal requires that any deliberation mechanisms produce for the agent a method of getting from its present state to the goal satisfying state within the solution search space (see Section 4.2)

iv. All reasoning must take note of constraints, and attempt to meet them, e.g. real-time hard and soft constraints, i.e. behaviour should be timely (see Section 2.1.2.b)

v. The achievement of a goal should be a process of interleaved planning and execution based on the agent's ability to predict the environment and the cost of doing so. In some situations it may be 'cheaper' to let the environment control action in a more non-goal mediated manner, i.e. via reflexive behaviours.

vi. The speed of the agent's responses to environmental events should reflect the number of goal forming events within that environment, i.e. as the urgency of a situation increases so too should the speed of the agent's responses.

vii. Given that as urgency increases agent response times must be reduced it can be seen that less time will be available for deliberation. In situations where the processes of deliberation are constrained they should still produce some form of behaviour, even if that behaviour is sub-optimal, i.e. an agent's deliberation processes should show graceful degradation under real-time constraints (e.g. Brooks 1986, Hayes-Roth 1990b). In such situations reflexive (type I and III - see Appendix A) behaviours provide a quick method of mediating behaviour.

viii. Where behaviours or goals are seen as being in conflict an agent's command and control system should take note of this for future reference (e.g. Hayes 1987).

3.7 Summary

In response to environmental events it is likely that an agent embedded within a real-world environment will be continually forming new goals, and that the achievement of any such goals (new or already extant) within the agent's command and control system will vary in importance to the agent with time, as determined by a variety of factors, not least:

i. The agent's current knowledge of the environment.

ii. The goals present within the system.

iii. The agent's currently active goals.

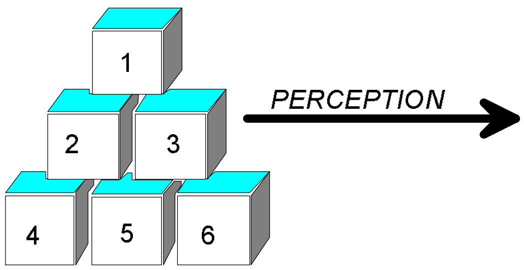
More importantly, because of the plethora of resource limits imposed upon an agent, it is likely that many goals will be in competition with each other for those limited resources. For this reason an agent embedded within a real-world environment must achieve an appropriate balance between the time that it spends deliberating about which goals it will attempt to satisfy (Task selection), how it will attempt to do so (Action Selection and Arbitration), and the time that it spends actually executing any resultant actions (Action Execution), i.e. a balance between the four points laid out within Section 1.2. This balance will change in response to the current state of the agent, the environment, and the agent's ability to predict the environment. It should thus be seen that the real-time performance of behaviour by an agent within a dynamic and complex environment is a problem of intelligent control (Hayes-Roth 1990a & 1990b). Finally, in keeping with the aim of this paper, such a balance should be achieved in a manner which allows for the integration of learning capabilities. The next three chapters provide an overview of the variety of approaches that have been posited over the years in an attempt to counter the problems associated with autonomous agent command and control.

4. TRADITIONAL AI

4.1 The Symbol System Hypothesis

When addressing the problems concerned with how to go about achieving an effective command and control system for an autonomous agent, the approach that has traditionally characterised AI has been to address the four points laid out within Section 1.2 via a mechanistic breakdown of a system into functional information processing modules (e.g. Brooks 1986, 1990, 1991; Maes 1991). Hence, up until quite recently the majority of command and control systems proposed have been systems generally modularised into mechanisms for perception, planning, and motor output, where the modules and the communications between them have been based upon the manipulation of symbols from an abstract symbolic set; the so called Symbol System Hypothesis. Such a hypothesis greatly influences the size and complexity of the design space that is open to an agent designer. Allow me to elaborate. A system based upon this type of command and control paradigm would, using a perceptual module, gather information from its environment and then attempt to transcribe that information into some form explicit domain representation; usually in terms of a set of primitive propositions (Lyons & Hendriks 1991), e.g. SITUATED CALCULUS (McCarthy & Hayes 1969); SNLP (McAllester & Rosenblitt 1991); STRIPS (Fikes & Nilsson 1971 & 1993), FORBIN (Miller et al. 1985), BUILD (Fahlman 1973). For example, a system might represent a tower of boxes with many propositional descriptors such as shown in Fig. 4.1 below.

Fig. 4.1 - A Symbolic Representation



on box1 (box2 box3)
on box2 (box4 box5)
on box3 (box5 box6)
on box4 (table)
on box5 (table)
on box6 (table)
on table (box4 box5 box6)
clear (box1)
unclear (box2 box3 box4 box5 box6)
size box1 (10 10 10)
etc.

All reasoning by the system, in terms of what to do next etc., would then take place upon this representation within higher level cognitive modules, for instance a planning module. The point to take note of though, is the fact that the representation(s) used would be based upon a symbolic set abstracted from the world and chosen by the system's designer not because of the sets affinity with the environment, but because of its utility within the higher level reasoning processes. As such, the abstract symbolic set utilised within such a system, being divorced from the actual domain, gains significance only in terms of the meanings assigned to it by either the system, the designer, or an observer, but not in terms of the domain itself. A good example of this is the use of everyday written language, e.g. newspapers, books, etc.

However, this may not be as debilitating as it appears at first sight. Indeed, systems that have attempted to do away with such an explicit representation, divorced from the immediate domain, have stumbled when trying to carry out abstract deliberation of the very sort which appears to characterise 'intelligent' human behaviour. For example, our ability to think about that which is removed temporally, physically, or even conceptually. Indeed, it seems improbable that such reasoning can ever be achieved without the use of some form of abstract representation of both the problem in question, and the domain the agent inhabits (see Section 6.2).

To summarise, the use of symbolic reasoning dictates that there must be mechanisms for the transcription of domain information into the abstract symbolic set that is to be used by the system, and that all reasoning mechanisms be able to utilise such a symbolic set. However, this leads to quagmire of problems:

i. What is the correct symbolic description of the world?

This is a question which has influenced both scientific (e.g. Green 1969; Hayes 1969; Brooks 1986, 1990, 1991a) and philosophical (e.g. Plato, Aristotle, Descartes, Kant - in Magee 1988) exploration throughout the ages. Indeed, many theories can be seen as being determined by the manner in which the information they attempted to explain was represented at the time of their conception. Take as an example, the difference between viewing entities with an aim to feeding as opposed to fighting. Both may describe the same entity with markedly different results in terms of the utility of the resultant information. Hence, the symbolic description an autonomous agent's command

and control system utilises to represent information is vitally important to the viability of that agent within its intended domain (e.g. Sloman 1994b & 1994c). Not least because an incorrect symbolic description may leave the agent in a position where it is unable to recognise certain types of events and entities (cf. Kant), or carry out certain problem solving tasks (Peterson 1992).

ii. Abstraction Level.

To what *depth* should we represent the environment. For instance, a chair may be represented as a unified object made up from a collection of smaller objects, such as four legs a seat and a back. In turn, each of these may be represented in terms of another collection of objects (if appropriate), and this process of progressive deepening may continue all the way down to an atomic level. Clearly, the level of depth (or abstraction) required when representing a chair within a symbolic representation is not a set thing, but one which dynamically changes in accordance to the situation an agent finds itself in, and the tasks it has set out to achieve (e.g. Kaelbling 1986; Fehling et al. 1989; Simmons 1991; Basye 1991). In many respects therefore, this problem mirrors that of the symbolic choice as outlined above within point (i) above.

iii. Representational Completeness.

One of the major problems with the use of the Symbol System Hypothesis is that it has tended to lead towards the production of general purpose visual systems that have been aimed at attempting to deliver a complete symbolic description of the world. Part of the thinking behind such systems being that if such a system could be produced then it could be slotted into any command and control system that required such a symbolic description. However, this approach at gaining environmental information assumes that:

a. The production of a full domain description in symbolic terms in computationally tractable, even in theory.

b. The symbolic description required will be domain and task independent.

With respect to point (a), as we have seen within Appendix E (Section 1) and within Section 2.1.1.b, Tsotos (1989 - from Chapman 1989) has shown that if vision were computed bottom up, i.e. without state, and without constraints on possible inputs, it would be a task of exponential complexity. However, the addition of a small amount of state can cut this exponential down to a linear function. In other words, vision as an entity which produces a full world model is impractical, but given this, which parts of the domain should be represented?

iv. The Frame Problem.

This above is closely tied with a problem that has been termed the Fame Problem, a term first coined by McCarthy & Hayes (1969) with the advent of Situated Calculus. That is when changes are made to a symbolic representation how *far* should those changes be propagated? Such a problem is clearly seen within, for example, the representation of a chess board as pieces are moved. When one piece is moved how many of the others become effected? If we only update the descriptions for those close by, do we then update the descriptions for the pieces close by each of those pieces, ad infinitum? It is interesting to note how Collins & Krulwich (e.g. Collins 1991a; Krulwich 1991a) used a form of attention in the guise of focus rules in an attempt to address this problem (see Appendix D Section 1).

v. Large search required in traditional planning.

The transcription of the world into a viable representation (ignoring the problems associated with what forms a good symbolic description of the world) gives rise to computational problems as well (e.g. Georgeff & Ingrand 1989; Korf 1987; Horvitz 1989; Basye 1991). As we have seen above, the use of an explicit symbolic representation of the world in which an agent is embedded can create a huge 'database' of information containing many hundreds, thousands, or even millions of entity descriptors. As we shall see later within Section 4.2 on Classical Planning techniques, planning has traditionally been thought of as the creation of a path, via search, through the space of possible world models. Given the size and complexity of such a space, it becomes abundantly clear that in all but the very simplest of domains, *blind* search, i.e. search that is unconstrained by any form of guidance, would be of exponential size (cf. the space of possible system designs).

This problem has necessitated the obvious need for methods by which the size of the search space can be pruned, e.g. evaluation functions, heuristics, or constraint posting. However, such pruning of the search space invariable means that some paths that are created will not be optimal, and indeed, may fail altogether. Yet, even with search space pruning planners spend most of their time creating, elaborating, and deriving properties of models of possible worlds. It is argued by some (most notable Brooks, Agre & Chapman, and Maes - see Section 6.1), that in concrete activity such explicit representation mostly just gets in the way. Avoiding representing irrelevant information collapses the state space of possible world models, and hence the computational problems associated with plan creation.

4.2 Classical Planning

The traditional view of a plan is a structure which has a starting step, a variable number of intermediate steps, and a final goal satisfying step, where each of these steps is specified in terms of its required input state, the operators it applies, and its resultant output state. Such specifying is normally done via the use of pre-conditions, actions, and post-conditions (respectively), where the conditions normally take the form of propositions that must be, or will be, true within the environment (Allen 1984). Accordingly, it can be seen that a goal (or a subgoal) is not a single state, but generally a property that is true of a number of states within the world (Korf 1987). However, referring to goals as sets of states is often awkward, hence a commonly utilised approach is to say that a world state satisfies a goal if the propositions concerning that goal are true within that world-state (Barrett & Weld 1994). Hence, this approach allows a simple quantification of world state to goal 'closeness' where the more goal propositions that are true within the world state the closer that state is to one which satisfies the goal. This is the main axiom that planning uses when attempting to solve goals. Using such a view, a complete plan is one in which the input state for one step matches the output state of the previous step, from start to finish, and the planning process is the achievement of this such that the final resultant plan is commonly a linear sequence, or an acyclic directed graph of actions (although the full range of programming control structures are theoretically available). The result of planning within this paradigm is that plans are commonly program like (Collins et al. 1991); the plans representing everything from high level goals to the system's executable actions. *Classically* the creation of such a plan is seen to be carried out in full prior to the execution of any part of the plan, and concomitantly, plan execution within such a system involves following through this program like plan without the need for decisions, i.e. classically plan creation and plan execution are two separate processes, one which involves deliberation and one which does not, e.g. BUILD, Fahlman 1973; TWEAK, Chapman 1985 & 1987 (Appendix B Section 6); CHEF (Appendix D Section 1), Birnbaum et al. 1991.

Let me elaborate. Planning has traditionally been defined as the creation of a path from an agent's initial starting state to its final goal satisfying state, via the application of system operators, and search through the N-dimensional search (state) space of possible world models. However, one of the main advances that Sacerdoti (1977) made was that it showed how a shift from searching through a space of possible world states to that of plan states, could drastically improve the efficiency of search (see Appendix B Section 4 & 5).

Analysis of this plan creation task reveals that there exist a number of parameters which determine its complexity:

i. The overall size of the possible state space.

Determined by the problem and the domain in which that problem is instantiated, but not by the agent's knowledge of either. As an example, the size of the possible state (or problem) space for a hiker trying to cross a lake is the size of the lake itself, regardless of whether the hiker knows that size or not. While this has some bearing on the length of any resultant plan, it has little effect on the optimality of that plan.

ii. Availability of system operators.

If planning is viewed as the creation of a path via the application of operators at each node in the problem space, the availability of <u>applicable</u> operators is at each node can be seen to have the following major influences on the manner in which the agent attempts to achieve its goals:

a. The branching factor.

The number of available paths leading from each node in the problem space is determined by the applicable operators available to the system. Hence, the fewer the operators that a system can apply the lower the branching factor. This has a bearing not only upon the length of any final plan, but also upon the time a system must spend deliberating about which path to go down. As an example think of the differences in taking a direct 'A' road route into town, and taking an indirect 'B' road route. The direct route is shorter, and involves fewer junctions than the indirect route making it quicker, and 'easier' to follow; fewer decisions need to be made as to which turning to take.

b. The applicable operator Search time.

A system may spend large amounts of time searching for applicable operators at each node, time which it may not have, and this has an obvious bearing upon the optimality of the planning process. It should be noted that this is not the same as the deliberation time required by a system in choosing a particular path once it has all the applicable operators. In an analogous way, the time that you spend finding out the various routes you <u>could</u> take to get from London to Glasgow, is not at all the same as the time that you take in deciding the route you <u>will</u> take, and this is different again from the time that you then spend taking that route! Each is separate, and each has a bearing on the overall optimality of the chosen solution, cf. search time for plan chunks within case-based planning (see Appendix B Section 7 & 8).

iii. The inter-nodal distance.

The availability of operators at each node in the problem space determines not only the number of choices that a system has open to it at each of those nodes, but also, by determining the 'distance' between the nodes within the problem space, the 'ease' of any resultant plan.

It can be seen from this that there are two main influences that impinge upon the efficiency with which a plan can be produced by a planner:

i. Search space size and complexity.

ii. Search Efficiency.

Given the above considerations, there is an obvious need for methods by which the size of the search space can be pruned, e.g. constraint posting, and search itself guided, e.g. via the use of heuristics, and other evaluation functions. Consequently, a number of techniques (detailed within Appendix B) have been developed to alleviate these problems:

i. Sub-goals.

ii. Independence.

iii. Serialisation.

iv. Abstraction hierarchies.

v. Partial order planning.

vi. Constraint posting.

vii. Macro operators.

viii. Case-based planning.

4.3 Problems With Classical Planning

One of the major problems with the classical planning approach to the design of autonomous agent command and control systems is that it makes three very large assumptions:

i. That the system has full domain knowledge.

The completeness of the search space, in terms of the nodes present and the inter-connections between them, is determined solely by the agent's knowledge concerning the domain in which it is embedded. For obvious reasons, the more complete this knowledge the greater the chance that problem solving search will:

a. Identify a possible path.

b. Identify the shortest and/or the most optimal possible path.

It is abundantly clear, that within real-world domains such completeness of domain knowledge is not practical both in terms of acquisition, and also in terms of storage. Thus, agent's embedded within real-world domains are always going to be knowledge poor, and hence it is always possible that problem solving search may 'overlook' something of vital importance to the satisfaction of goals, i.e. search can not guarantee that the plans it creates will be both goal satisfying or indeed optimal. They cannot prove completeness.

ii. That the domain is fully predictable.

As shown within Section 2.3, since it is generally the case that the complexity and dynamism of any real-world domain precludes full predictability, search can never by optimal. Furthermore, the results of such search cannot then be guaranteed to hold true long enough from the system to fully traverse any created plans anyway (e.g. Lyons & Hendriks 1991; Firby 1987; Hanks & Firby 1990). Because a real-world domain is inherently unpredictable, the time an agent spends in 'deliberation' will often reach an optimum point beyond which the benefits of increased search completeness will reduce. This point will be based not only upon the predictability of the domain, but also upon the goal in question. Hence, a planner embedded within a real-world needs to balance the time that it spends creating and then searching a problem space, against the dynamism of the domain, and the goal in question - a decidedly non-trivial problem.

iii. That deliberation time is unconstrained.

> Another key assumption made use of by classical planning systems is that no constraints are placed upon the time that they spend in deliberation. This is clearly not the case given the multitude of time constraints that will exist within any real-world domain (see Section 2.1.2.b). Classical planning systems are notoriously slow because of the manner in which problem solving is achieved within them, and for this reason alone they do not provide a good approach for the design of an embedded autonomous agent's command and control system.

Because of these assumptions, classical planning systems all suffer from the major problem of being shackled by implementation within very simple domains, especially the Blocks World. The trouble is that this clearly does not present the systems with pressures that are realistic. Although some attempts have been made to rectify this problem, e.g. using slightly more complex versions of the domains (e.g. Fahlman 1973), or different domains altogether (e.g. CHEF - see Appendix D Section 1), these attempts do not hide from the central unrealistic nature of classical planning.

4.4 Reactive Planning

The adoption of the *classical* modus operandi over the years by the AI community has given rise to a set of conventions concerning the inputs, outputs, and the sorts of things that thought and reasoning do, and these do not seem to be capable of accounting for large aspects of what goes into intelligent autonomous behaviour (Brooks 1990). Probably the largest reason for use of classical techniques within simplified domains was the inadequacy of computational resources available for use by researchers. However, such a reason does not detract away from the essential problems of the approach. To counter this, and in an attempt to circumnavigate the inherent problems associated with 'classical' planning techniques, there has been a gradually shift in the focus of research over the past 10 years towards a more situated view of activity, systems being designed to demonstrate less sophisticated tasks, but operate robustly within complex, real-world domains,

The term 'situated activity' is used here to cover both reactive planning systems (described below), those which use explicit symbolic representations and are internally structured into various functional modules, e.g. for planning, and Nouvelle AI systems (described within

Chapter 6), those which don't use explicit representations and/or are internally structure into behaviour producing modules, e.g. for walking.

The so called Reactive Planners, e.g. Nilsson (1988), FORBIN (Miller et al. 1985 & Dean et al. 1988), RAPs (Firby 1987)[7], PENGI (Agre & Chapman 1985 & 1987)[8]. These systems avoid the computationally expensive problems associated with deliberating about complex, dynamic, and uncertain domains by making action selection decisions at execution time based solely on the current environmental circumstances rather than on the results of any type of deliberation, e.g. planning. Essentially therefore, they have attempted to reduce the combinatorial costs involved in planning within real-world environments either by removing planning per se, or by pre-compiling plans into the system, i.e. the term 'reactive planning' is oxymoronic in that no planning actually takes place. The best example of this is the Universal Planning paradigm (Schoppers 1987)[9] in which perceptual events are transcribed into motor output via the use of a large look-up table without any recourse to higher level deliberative processes.

Reactive Planning systems leave a lot unaccounted for, not least their myopic methods of controlling non-goal directed behaviour (e.g. Firby 1987; Lyons & Hendriks 1991; Gat 1991), and their avoidance of questions concerning the command and control of situated activity via the displacement of planning problems on to the system's designer.

4.5 Summary

In summary, the following of the Symbol System Hypothesis within designs for an autonomous agent's command and control systems has allowed, in theory, a degree of domain independence, but at the expense of solution optimality. Furthermore, reasoning techniques within traditional AI have proven to be computationally expensive to the point of being exponential and thus intractable for many large problems. This is clearly shown in the techniques and implemented systems which fall under the Classical Planning heading, e.g. BUILD, Fahlman

---

[7] See Appendix C Section 2.

[8] See Appendix E Section 1.

[9] See Appendix C Section 1.

1973; HACKER, Sussman 1975; NOAH, Sacerdoti 1977; and TWEAK, Chapman 1987. In an attempt to address such problems one approach used by proponents advocating the use of abstract symbolic sets within autonomous agents, has been to make those agents more reactive, doing away with large amounts of deliberation and its associated problems, e.g. Brooks's Subsumption Architectures (1986)[10], Agre & Chapman's PENGI (1987), Pfeifer's Fungus Eater (1992). Such an approach has obvious drawbacks. The cause and effect conceptualisation of agent control leads to myopic behaviour, behaviour which cannot take advantage of information that is either physically (Gat 1991), or temporally (Firby 1987) removed, and fails to adequately explain behaviour that persists in time (e.g. driving). It can be seen that both approaches, classical and reactive planning, offer a simplification of the planning task, where classical planners typically inhabit simple static domains, but carry out detailed deliberation, and reactive planners typically inhabit complex dynamic domains, but do away with much deliberation (Pryor & Collins 1991).

The integration of reactive and classical techniques is viewed by many as a means of producing truly situated agents, able to react, as well as deliberate. The myopic tendencies of reactive planners countering, and in-turn being countered by, the more costly deliberative approach of classical planning (although, it is interesting to note that Chapman (1989) takes the opposite stance, preferring to see such a hybridisation as combining of each approaches worst aspects!). The following chapter, thus looks at embedded agent command and control systems, that integrate reactive and classical planning to produce truly situated activity.

---

[10] It should be noted that the work done by Brooks' while falling under the general heading of a reactive planner, posits the absence of explicit symbolic representations. As such it is covered within Chapter 6, on Behaviour-Based AI.

5. INTEGRATED ARCHITECTURES

It would appear that the planning paradigm an agent follows should be determined by the its ability to predict future events within the environment, the completeness of the its domain knowledge, the constraints that impinge upon it, and the computational powers available to it. The paradigms of classical, or reactive, planning rely on the fact that the environment is either fully predictable, or fully unpredictable (respectively), extremes which are rarely the case. It is much more likely that real-world environments will fall somewhere upon a continuum between these two, and hence it follows that an agent embedded within a realistic environment should follow neither paradigm in complete exclusion to the other (e.g. Hammond et al. 19xx; Horvitz 1989; Hanks & Firby 1990; Gat 1991; Collins et al. 1991; Hayes-Roth 1990a & 1990b; Pryor & Collins 1991). Furthermore, as mentioned earlier within Chapter 2, an agent embedded within a real-world environment is likely to suffer from a number of constraints which limit the number of events that it can 'deal' with at any one time, i.e. the agent is likely to form multiple conflicting goals, whose importance to the agent dynamically varies with respect to time (Sloman 1987, Beaudoin & Sloman 1993). Thus, an agent must balance between:

i. Deliberating about which goals to deal with.

ii. Deliberating about how to deal with those chosen.

iii. Executing resultant plans.

iv. Information gathering

Such a balance being determined with respect to the agent's ability to predict future events, the constraints placed upon the agent, and the goals in question. Agent behaviour must be seen to arise not simply from the dumb following through of precompiled plans (e.g. Universal Planning), or from simplistic reflexive type behaviour (e.g. Pengi), but rather as an active, intimately integrated process of generating goals, deliberating, and effecting. It can be seen that integrated approaches to agent command and control do address the means of achieving this balance, but not without suffering from a number of shortcomings.

5.1 Model-Based Reasoning

As has been expounded within the previous chapter, the shift in focus for an agent's command and control system, away from abstract planning towards situated activity, means that the majority of that agent's components cannot now be viewed as simple 'program' creators and interpreters. Instead, they must be seen as a more articulated set of components each devoted to controlling some aspect of behaviour, and hence allowing for the same type of specialisation and thus increases in efficiency seen in factory production lines, and so offering benefits over more general purpose components.

In such a system, a large proportion of the responsibility for the system's performance efficacy is seen to shift towards the run-time planning and execution components, rather than being simply on the planning components. Consequently, it follows that the blame for performance failures must also shift more on to those components, rather than lying with the plans themselves. The added advantage of this method of failure driven performance improvement is that errors that occur because of faults within a component are likely to lead to repairs that have the broadest applicability in terms of other system behaviours, so long as the repaired component is used by other system components.

Since the tasks in which a system engages are not completely disjoint and their interactions, e.g. between generating goals and plans, require that the system's components be correspondingly intertwined, an agent's repair/learning component must be able to reason about the interactions occurring between components when formulating new concepts concerning the reasons behind plan failure. Furthermore, any such concepts that a system learns, e.g. fragile objects should not be dropped, must depend upon its perceptual and plan execution abilities, the plans that it typically generates, and the constraints under which it operates. In view of this, the approach Collins, Krulwich, and Birnbaum (Collins 1991a, 1991b, 1992; Krulwich 1991a, 1991b; Krulwich et al. 19xx; Birnbaum et al. 1991) take to the achievement of situated activity is the use of model-based reasoning. They have utilised this within two systems CHEF and CASTLE as detailed within Appendix D Section 1.

Model-based reasoning can be seen as providing a unified approach to the tasks of plan debugging and the learning of new concepts that arise from such debugging. Predictions concerning the expected behaviour of a system are derived from an explicit planner self-model of that system in conjunction with the current situation and the goal being planned for. Upon performance failure these predictions are then used to determine why errors occurred, and to develop the concepts that must be learnt in response to such failures.

The key issue in model based debugging is inferring the faulty assumptions underlying an observed symptom. Thus, performance expectations are generated by inferences that are stored within *Justification Structures*. Upon performance failure, these are tracked through, allowing the system to identify and isolate components that were at fault. For each identified component a *Component Performance Specification* describing the correct behaviour is then used to identify correct behaviour that wasn't present, allowing the system to use a deductive inference engine to construct an explanation of why the failure occurred. Such an explanation is generalised using EBL (explanation-based learning). The most general operational expressions are collected within a *Proof Tree,* whose leaves form the antecedents of new rules to embody the concepts that must be learnt in response to the failure. Hence, not only does Model-Based reasoning[11] provide a paradigm for fault diagnosis it also provides a basis for a theory of repair (Collins et al. 1992).

It is important to remember that because it is the component that is 'fixed' rather than any specific plan, the new rules given to a component after a failure are transferable from that particular failure situation to others. Hence, it is the view of Collins and Krulwich that a theory concerning domain-independent planning can be achieved only by adopting a position that it is a theory about 'how to learn to plan' in new domains, rather than simply a theory on 'how to plan' in new domains. Within such a theory, the interaction between the system's knowledge concerning planning and its experiences within domains, generates the detailed, domain specific knowledge required by the system for effective domain-independent planning in a similar manner to 'critics' within HACKER (Sussman 1975). The efficacy of the system thus becomes the responsibility of the system itself, rather than of the system's programmer(s). Additionally, the use of self-models within embedded agents has a number of very important benefits, not least:

---

11 As far as I understand, EBL differs from Model Based Reasoning (MBR), in that EBL uses a list of the decisions that were made prior to some outcome to decide using MBR where a possible problem occurred.

i. They are biologically plausible.

That's to say that they appear to offer a good analogy to some of the reasoning processes that occur within humans and possibly other 'intelligent' animals. Indeed, faults in the applied focus rules can account for faults in attention as experienced by humans, e.g. novice chess players, not noticing attacks.

ii. Emotional Behaviour.

Not only can a self-model be used in an attempt at modelling what another similar system may be thinking, but the results of such modelling may lead to beliefs, whether correct or not, that initiate emotional responses, e.g. anger at a goal thwarting incident perceived as deliberate. Wright (1994a) is currently looking at how a model of another agent, used to make predictions about that other agent, may give rise to inappropriate perturbant states. Such states arising when that agent no longer exists and the model cannot be easily removed, or more subtly when the model used is inaccurate. Furthermore, Wright is also examining ways in which self-models may be used to manage such emotional episodes.

iii. Meta planning.

For a system to plan effectively within any domain, it should have knowledge concerning its own strengths and weaknesses. Such knowledge can be used to influence how planning takes place, what actions the system can or can't do, e.g. can you jump the gap? Etc. Furthermore, knowledge concerning the abilities of various sub-systems, e.g. perceptions, can influence the types of plans produced, e.g. if you are deaf look more carefully than normal when crossing a road.

The BDI (Belief Desire Intention) formalism, as discussed below, provides an alternative approach to situated activity, viewing such agents as behaving in accordance with a notion of rationality.

5.2 The BDI Formalism

An agent must not only carry out the processes of means end reasoning, but also weigh up competing alternatives on the basis of subjective expected utility, i.e. incorporate a degree of decision theoretic control (Bratman et al. 1988). In accordance with this, decision making processes should be viewed as being influenced by a number of mental attitudes, *beliefs*, *desires* and *intentions*. These represent respectively, information states concerning the environment, evaluations of those states, and past decisions central to the adequate, or optimal, performance of an agent when it is resource bounded (Rao & Georgeff 1992). Designs that adhere to such an approach tend to be termed BDI architectures and a number of attempts have been made to formalise such notions, most notably Cohen & Levesque (1990[12]) and Rao & Georgeff (1991).

Central to Rao & Georgeff's (1991) view of rational behaviour is their use of a formalism similar to Computation Tree Logic (CTL) for representing the world as branching time trees. Points on these trees are called *situations,* branches represent choices that are available at each moment in time, and *events* transform one point in time into another. Associated with each situation are a set of *Belief Accessible Worlds* (BAWs), i.e. those worlds that the agent believes to be possible, where each of these are themselves time trees. For a belief to be adopted as true in the current situation it must be true in all of the BAWs leading from the agent's current situation, and as such situation branching into multiple time trees represents a lack of knowledge on the part of the agent - a representational extension that most conventional models of belief do not support. Converse to this, branching within time trees represents the choices available to the agent for action selection.

Similarly, also associated with each situation are *Goal Accessible Worlds* (GAWs), and *Intention Accessible Worlds* (IAWs), sub-worlds which correspond to increasingly selective choices by the agent in terms of its desire for, and commitment to, possible courses of action. GAWs represent the goals of the agent, which unlike an agent's desires must not only be mutually consistent, but also believed by the agent to be theoretically possible (even if it doesn't believe that a point in time will ever be reached where such achievement will occur). Again, like beliefs, for a goal to be adopted it must be true in all the GAWs leading from the agent's current situation. The last structure associated with situations are IAWs representing those

12 In Rao & Georgeff 1992.

GAWs that the agent has committed to realising, i.e. those worlds that represent the agent's commitment to certain desired future paths.

The above formalism differs from that of Cohen & Levesque in two important ways. Firstly, it supports a notion of BDI interaction such that an agent who intends to do a certain action should not be forced to intend all of the associated side effects for that action. For example, an agent can believe that pain inevitably accompanies going to the dentists, but still have the goal (or intention) to go, without the having the goal (or intention) to suffer pain. Secondly, it treats intentions as a basic 'attitude', shifting the emphasis of future commitment, and behavioural coherence, away from the definition of intention on to the processes of intention maintenance and revision. Consequently, different agents can have different strategies for intention revision, or *commitment strategies*, which in turn give rise to different types of global behaviour. For example, an agent which is blindly committed to its intentions will maintain them until they are believed to be achieved, giving rise to very 'single-minded' like behaviour. Conversely, an agent which is said to be open minded, maintains its intentions only so long as they are still goals. Hence, such an agent will not only drop intentions when they are no longer believed possible, but also at the suggestion of other agents.

In summary it can be seen that the formalism gives a unifying approach to embedded reasoning, as the world changes so too do the structures that influence reasoning, and hence so does the agent's behaviour. However, it fails to address a number of important issues:

i. How a system may generate new goals.

ii. How intentions may be formed via deliberation.

iii. How intentions may be modified in the light of changing circumstances or reconsideration.

iv. How learning can affect BDI interaction.

v. How when more than one action is possible a single one may be chosen by deliberation.

vi. Most importantly, in terms of actual agent design, the above formalism is abstract, it does not address how such BDI systems could be implemented (Rao & Georgeff 1992).

Both points (v) and (vi) are addressed by PRS (Georgeff & Ingrand 1989)[13], a system which instantiates the BDI formalism within a real world embedded system aimed at maintaining a balance between execution and deliberation. However, while PRS is a flexible reasoning system, it fails to account well for the creation of goals, their effective management, and the effects of behaviour on stored information. The first two points concerning goal generation and handling are eloquently addressed by NML1 (Beaudoin, 1994) which extends PRS with a richer goal definition and more clearly defined management processes.

5.3 NML1

Based on PRS, NML1 (Beaudoin 1994) addresses the majority of the problems that impinge upon PRS and the basic BDI formalism, providing support for the generation and management of goals (points i, and iii in Section 5.2 above) within a resource bounded agent. In so doing it provides not only a richer goal representation, but also more clearly defined goal manipulation mechanisms than PRS and many other systems. Furthermore, the concept of meta-level control is investigated and extended from that of KAs in PRS to specific management processes. NML1 consists of a number of structures concerned with vigilation, management, and meta-management, and below is a brief summary of these areas:

5.3.1 Vigilation Mechanisms

Acting in accordance with the system's current control strategies and activities, information is gathered from the environment via perceptual processes under the auspices of a Perceptual Control Component. The information so gathered is collated and stored in the form of an explicit symbolic world model monitored for firing conditions by a potentially unlimited number of Goal Generactivators. In keeping with the idea that an agent must remain responsiveness to environmental events, Goal Generactivators

13 See Appendix D Section 2.

not only act asynchronously and in parallel with each other, but also with the rest of the system. They express agent concerns, such as eat food, and are capable of re-activating suspended goals as well as generating new ones - both in response to internal as well as external events.

Given the goal generating potential of the above, it is imperative that an embedded agent maintain control over the flow of newly generated goals, protecting resource-bounded processing from unnecessary computation and interruption. This is achieved via the use of an idiosyncratic dynamic threshold filtering mechanism, the Attention Filter, through which newly generated goals must pass. Consequently when new goals are generated they are proposed as candidates for filtering, either failing and being suppressed, or succeeding and 'surfacing' into Management Mechanisms. However, being dynamic the threshold may alter in accordance with the agent's 'busyness', and being idiosyncratic the filter can be instructed to always suppress or surface goals of a certain type, i.e. while computationally cheap, filtering is relatively flexible. One problem with such computationally cheap heuristic filtering, however, is that it is liable to make errors, allowing unimportant goals to surface while suppressing important ones. Although, it is argued that the benefits outweigh the costs, without an implementation this is difficult to judge, and rather than being constant is likely to be dependent both upon the agent's current goals and the agent's current environment. In summary therefor, although not infallible the Attention Filter should be seen as one of the key methods by which NML1 manages the flow of goals from perception into management.

5.3.2 Management Mechanisms

Upon surfacing into management, a number of processes must be carried out on a goal prior to any behaviour being directed towards its realisation. Not only must it be assessed for adoption as an intention, but it must also be scheduled for execution, and plans determined for it. These tasks are carried out by an interpreter invoking applicable management procedures, procedures which are considered to be independent processes running on dedicated processors. In this respect they differ markedly from the application on KAs within PRS. More importantly, NML1 offers the realisation that management processes cannot occur in a serial fashion. Often information required by one task is not available until another task has started to occur, and that task may only be able to execute fully when the first has started to return some result. In keeping with this, unlike in PRS where management is seen to occur serially, NML1 supports the cyclical partial execution of management processes. Finally, once a management process is in operation it may dispatch commands to an effector module for the execution of system actions in the agent's environment, or alter the manner in which the systems management operates. Thus, the processes of management refer to goal economy once a goal has surfaced (Sloman 1994e).

5.3.3 Meta-Management Mechanisms.

Within NML1 not only may goal-directed tasks such as scheduling, planning, or execution, be carried out, but so too may processes that refer to the management processes required to achieve these. As in PRS, decisions concerning the priorities of, or operation of, management tasks are termed meta-management processes, and often take a recursive definition which may show infinite regress. For example, NML1 may often need to decide that decisions concerning goal adoption, goal scheduling, or goal expansion, should occur at a later point in time - much like when we say "I'll think about whether/when/how to do that goal later". Furthermore, such meta-management allows a system to determine when management tasks have entered problematic states, such as 'thrashing', whether all the relevant issues concerning a goal, or set of goals, have been considered, or whether the timeliness of response is adequate. In the latter case, meta-management can effect the system's responsiveness not only by altering the mode of operation that management takes, but also by raising (or lowering) the threshold of the attention filter, or even deciding to suspend (conditionally or otherwise) previously active goals. Thus, where PRS alludes to meta-level control, NML1 defines it.

In summary, NML1 benefits from all the advantages of PRS, such as interruptible management processes, and interleaved (or concurrent) planning and execution, while also extending goal and meta-management functionality. However, NML1 does not address itself to efficient memory storage and retrieval, nor the problems associated with integrating learning into such a system. The BB1 and GUARDIAN systems described below can be seen to extend NML1s notion of time management, and management control strategies, while remaining within the BDI characterised approach of embedded computation.

5.4 Dynamic Control Reasoning

Hayes-Roth (1987) has shown that control reasoning can lead to an increase in behavioural efficiency when more than one action is possible at any one moment in time, and the agent is able to either alter the number of, or costs of, those actions. As such, the use of control reasoning should be seen as critical to the viable behaviour of an embedded agent, and has been investigated by Hayes-Roth within her BB1 and GUARDIAN systems (see Appendix D Section 3).

The BB1 Blackboard Control Architecture (Garvey, Cornelius & Hayes-Roth, 1987) provides an opportunistic, goal driven system capable of supporting both problem solving and control reasoning in a time sensitive manner (although doesn't support any form of learning). This is achieved via the use of two types of system knowledge encoded within functionally independent Knowledge Sources (KSs). These co-operate to solve problems via the recording and modification of a global data structure known as the blackboard, and are:

i. Domain KSs that encode problem solving information for actions on the domain blackboard.

ii. Control KSs that encode control plan information for actions on the control blackboard.

In this respect BB1 has much in common with the provision of Knowledge Areas (KAs) and meta-level KAs within both PRS and NML1. However, where BB1 differs is in its use of the above mentioned control plan blackboard in addition to the more normal domain blackboard. This control plan encodes information central to the manner in which BB1 produces behaviour, and should thus be seen as providing the system with a mean of control reasoning. However, while the control plan allows for a degree of reasoning that is above the norm, the production of behaviour within BB1 occurs in a simplistic fashion. Triggered KSs, regardless of their type, generating KS Activation Records (KSARs) which must then compete for execution in the following manner:

i. As a result of action execution, environmental events, or cognitive events, changes may occur in both the domain and control blackboard that lead to the satisfaction of KS trigger and precondition requirements. Such executable KSs are identified by the Agenda Manager which generates new KSARs for these, adds them to the agenda, and then rates them against the current control blackboard.

ii. The Scheduler chooses the highest rated KSAR from the agenda for execution by the interpreter.

iii. The Interpreter executes the action specified by the scheduled KSAR, changing the appropriate blackboard.

In summary, BB1 provides a uniform mechanism for both problem solving, and control reasoning within an embedded, resource constrained agent, and has been implemented within a number of systems. Best developed of these has been the GUARDIAN system (Hayes-Roth 1990a, 1990b, 1991a, 1993 - see Appendix D Section 3), primarily implemented as a system for intensive care monitoring, although recent extensions have been used to study the role of 'motivation' in behaviour production (Morignot & Hayes-Roth 1994), as well as methods for evaluating integrated architectures (Hayes-Roth 1991b). GUARDIAN extends dynamic control planning, as implemented within BB1, through the use of the Control Plan, a temporally organised pattern of control decisions, presenting the classes of operations that the agent intends to perform over some period of time, as well as the manner in which such performance is to take place. However, neither BB1 nor GUARDIAN are without their faults, the most crucial of these being concerned with:

i. The provision of learning mechanisms.

   BB1 does not provide support for any form of learning. This is supposedly redressed within the GUARDIAN system, which it is claimed, does provide learning mechanisms in terms of learning new factual information and new reasoning operations. However, no information is given as to how these mechanisms are implemented.

ii. The provision of memory management mechanisms.

Within GUARDIAN the Agenda Manager is non-exhaustive, and as such, the order in which executable operations are identified is critical. It is proposed (Hayes-Roth 1990, 1993b) that the Agenda Manager apply a heuristic best-first algorithm in its search to maximise the speed of identifying the best available operations. However, this appears to be a somewhat 'cock-eyed' idea. The expense in attempting to identify the best executable operation is not in rating those tasks, but in searching for them. Since the best-first algorithm must search the same search space as the Agenda Manager it is unlikely to result in any increase in identification speed. Indeed, surely such an algorithm would result in a decrease in speed because it must search the whole space of available operations prior to heuristically determining the order in which to select them for rating.

Given this, along with the claim that GUARDIAN accommodates a degree of learning and hence system knowledge is unlikely to remain constant, the issue of memory management which has been neglected must be addressed. Indeed Hayes-Roth (1991a) writes that "GUARDIAN does slow down over time as lisp machine memory fills up" (p. 81). This is important because it means that GUARDIAN is unable to guarantee its response times unless it restricts identification time (e.g. Hayes-Roth 1993b), and this may lead to the best executable operations remaining unidentified and thus sub-optimal agent behaviour.

iii. The provision of reasoning operation selection methods.

While GUARDIAN does support multiple methods for achieving the same goal (Hayes-Roth 1991a) it is unclear how those methods are chosen between.

The AIMAE architecture proposed within Chapter 7 addresses all of these points.

5.5 Summary

The above chapter has focused on model-based reasoning, BDI formalisms, and dynamic control planning. These approaches provide many mechanisms for the achievement of situated activity, but also suffer from a number of problems. Most notably, the abstract BDI formalism fails to provide a clear definition on how goals should be generated, how intentions may be formed and modified, how learning is to be supported, or how action selection and implementation are to take place. While PRS goes someway towards alleviating that last two of these problems, it still suffers from poorly defined goal generation and handling mechanisms, as well as a lack of learning. In response to this, NML1 has extended the goal management aspects of PRS, but still fails to include support for learning. Finally, extending the management of goals even further, BB1 and GUARDIAN both provide support for time management via the use of dynamic control reasoning, an important aspect of situated activity. However, as with the other systems, the management of memory is sidelined and learning, although claimed to be supported, is vague.

In summary therefore, PRS, NML1, BB1 and GUARDIAN, do move someway towards situated activity agents, however, they fall short most crucially in terms of learning abilities and memory management. The following chapter outlines a number completely different approach to the design of both an agent and its command and control system, eschewing the use of explicit representational schemes, and/or the modularisation that has characterised both classical and reactive systems.

6. NOUVELLE AI

As we have seen from previous chapters, investigation into the design of autonomous agent architectures has traditionally been characterised by two key points:

i. That action is causally mediated via the creation and manipulation of explicit symbolic representations.

ii. That command and control systems are decomposed into functional information processing modules.

However, alternatives to both of these (individually and together) have been proposed by various researches. This chapter aims to examine such alternatives along with a number of illustrating systems.

6.1 Issues Concerning Representation

One of the key characteristics of traditional AI is its utilisation of abstract symbolic representations in the causal mediation of behaviour - the so called Symbol System Hypothesis (Section 4.1). Such a design stance requires that an agent's command and control system have mechanisms not only for the transcription of domain information into such representations, but also for their subsequent use in the causal mediation of action. Both of these requirements come heavily loaded with problems, problems which a traditional theory of agent behaviour must address. For example:

i. The choice concerning which symbolic representations to use.

ii. The choice concerning the level of abstraction the world should be represented to.

iii. The degree to which representation should be independent from deliberation.

iv. The frame problem.

v. The large amount of search required for symbolic reasoning.

Because of the large philosophical and practical issues these points raise, a number of researchers have decided to eschew the use of explicit symbolic representations, preferring instead representations that are grounded totally within the here and now of the real world (e.g. Brooks 1986; Agre & Chapman 1987; Chapman 1989; Maes 1990a). This has been termed the Physical Grounding Hypothesis.

Proponents of this approach argue that without explicit symbolic representation the need (and ability) for a system to utilise abstract symbolic sets in the production of behaviour is eliminated. Such an elimination not only removes many of the problems associated with symbolic sets, but in as much as domain rules make many theoretically possible inputs impossible, allows the exploitation of domain structure to exponentially reduce the computational costs associated with perception (Chapman 1989). Consequently, the world is viewed as its own best model with explicit representations being both superfluous and costly.

Although a number of systems have been proposed that utilise physically grounded representations in the pursuit of situated activity, of landmark importance is Agre & Chapman's Pengi system (1987)[14]. Unlike any of the other Nouvelle systems, Pengi utilises physically grounded representations, but within a more traditional architecture. As such it provides a bridge between traditional AI systems and the more 'extreme' Nouvelle AI systems that do away with the traditional approach to architectural design as well as explicit symbolic representations.

The flip-side of this is, of course, that the removal of abstract symbolic representations from an agent also removes that agent's ability for abstract reasoning. However, it is hypothesised that analogous to the presence of structure within the relationship between the domain and perception, is the presence of structure in the relationship between perception and activity. Indeed, Ginsberg makes this point himself in arguing for planning: "Our environment behaves in predictable ways..." (Ginsberg 1989, p. 43). As such, describing and exploiting such regularities replaces abstract reasoning and provides most of the work that goes into designing a situated activity system (Agre & Chapman 1989). This leads to problems which are, if not unique, not generally those which effect more traditional systems.

---

[14] See Appendix E Section 1.

6.2 Problems With The Physical Grounding Hypothesis

Utilisation of the physical grounding hypothesis introduces a number of problems for an embedded autonomous agent - problems that may prove the stumbling-block over which the Nouvelle AI paradigm falls, these being:

i. The world is its own best model.

This premise creates problems in terms of the agent being able to sense its domain both frequently enough, and appropriately. It can be seen that the frequency of sensing is a function of the dynamism of the domain at the level of abstraction it is being observed at. Compare, for example, the position of a table over time with the position of a moving car over time, or more subtly, a flea and a chimpanzee inhabiting the same physical and temporal locations but living in 'different worlds'.

ii. Exploiting domain regularities leads to purposeful behaviour.

The premise that the world is its own best model, and that a system can produce purposeful behaviour by exploiting regularities within that domain holds a number of implicit assumptions:

a. That regularities exist within the domain.

This is misleading. Regularities within the domain are not dependent upon the domain per se, but rather upon the system that is attempting to utilise them in understanding and predicting domain events (cf. Kant[15]). As such, domain regularity is a function of the level of abstraction used to view the domain, the system's conceptual apparatus, and the computational resources available to that system for (what might be termed) predictive, or analytical reasoning.

b. That the agent is actually able to exploit domain 'regularities'.

---

15 From *The Great Philosophers* (Magee 1988 - Publishers Oxford University Press).

It is one thing to discover, or know, that a certain regularity exists within a domain, but it is quite another to make use of that regularity in the achievement of any given goal. Furthermore, certain regularities may be too expensive in terms of resources, e.g. time and money, to make their use a viable proposition for the system. For example, knowing that during the course of a storm there is likely to be electrical discharges in the form of lightening, is not at all the same as being able to exploit that regularity with the aim of harnessing electricity. However, it is predictive of ill health if you are caught in the open!

c. That perception is the only source of information.

The information content of any situation that a system finds itself in is not only dependent upon the agent's immediately sensed domain, but also upon its own internal environment, such as beliefs, desires, current goals, etc. (e.g. Wright 1994b). Indeed, Pryor (1991) argues that one source of information that is available for utilisation by a situated activity command and control system is that produced by its own reasoning processes. Of course, without any reasoning processes this source of information is removed from the action selection decision, and such a loss can only be at the expense of a system's ultimate success. As the saying goes, knowledge is power!

iii. Representations are physically grounded

In attempting to reduce the computational problems associated with explicit symbolic representations, the use of representations that gain significance only in terms of the here and now, leads to a number of problems:

a. The conceptualisation of cause and effect.

Use of representations that gain meaning only in terms of the here and now leads to a cause and effect conceptualisation of behaviour which fails to adequately explain behaviour that persists in time, e.g. driving.

b. Reasoning abilities.

It seems likely that 'basic cognitive capabilities' can be, and probably are, even in humans, based upon mechanisms that do not require explicit symbolic representations. However, 'complex cognitive capabilities'[16] that require an agent to reason about entities from which it is removed, either in terms of space or in terms of time, would appear to require abstract symbolic representation(s) divorced from the immediate here and now. Indeed, such reasoning can be viewed as the determiner of whether an agent is intelligent, or not, and this would explain why it is so difficult to determine whether an animal is intelligent; the judgement would require impossible to gain knowledge regarding whether that animal used a symbolic representation of its environment, and was able to reason using that representation.

Without an explicit representation of the domain the system is denied the ability to reason. Although Agre & Chapman (1987) claim that such reasoning is superfluous in the command and control of situated activity, this is flawed by the fact that without the flexibility accorded by such reasoning the system can never:

i. Reason about the temporal aspects of a domain, situation, or problem.

ii. Reason about which goals to postpone in favour of others.

c. Learning abilities.

Although, a minority of the systems utilising the physical grounding hypothesis can be said to carry out limited forms of learning, none of them allow more explicit learning. For example, the type of learning which would allow a system to acquire information from one part of the environment for use within another part of the environment, either temporally, or physically. Furthermore, without using symbolic representations it is difficult to visualise how any Nouvelle AI system could learn complex relational facts such as these.

---

[16] The term *basic cognitive capabilities* is used to describe information processing systems that are placed at a low level in the hierarchy of control within an agent's command and control system. For example, data-driven perception, physiological reflexes, and the control and co-ordination of effectors. The reverse is true of *complex cognitive capabilities*. For example, goal scheduling, and planning.

iv. No notion of a goal.

Without symbolic representation there can be no explicit notion of a goal, and thus the mechanisms aimed at dealing with multiple domain demands will always be 'simplistic' and 'myopic' compared to systems that can make use of goals. For instance, while environmental demands can be prioritised without a means of representing them they can not be scheduled. Furthermore, I am doubtful that such prioritising is possible in the first place, given the marked disability such systems exhibit for reasoning.

v. No notion of time.

This is very important. Without a notion of time, goals can not be postponed until later, nor can the effects of real-time constraints be taken in to account in the selection and generation of behavioural actions.

It can easily be seen from the above that the use of physically grounded representations within a system carries large burdens in terms of a plethora of associated problems.

6.3 Issues Concerning Agent Structure

The approach taken by traditional AI in the structuring of an agent's command and control system has typically been to divide such a system into functional information processing modules. For example, modules that deal with perception, goal scheduling, plan deliberation, learning, and action execution. Such an approach leads to designs where no one module can produce behaviour on its own - behaviour being a product of all the modules in combination - and where improving the system's performance efficacy requires the improvement of already extant individual modules (Brooks 1990). In contradistinction to this is the Behaviour-Based approach, best implemented within Brooks's Subsumption Architectures[17], Kaelbling's Situated Automata[18], and Maes's Spreading Activation Networks[19].

---

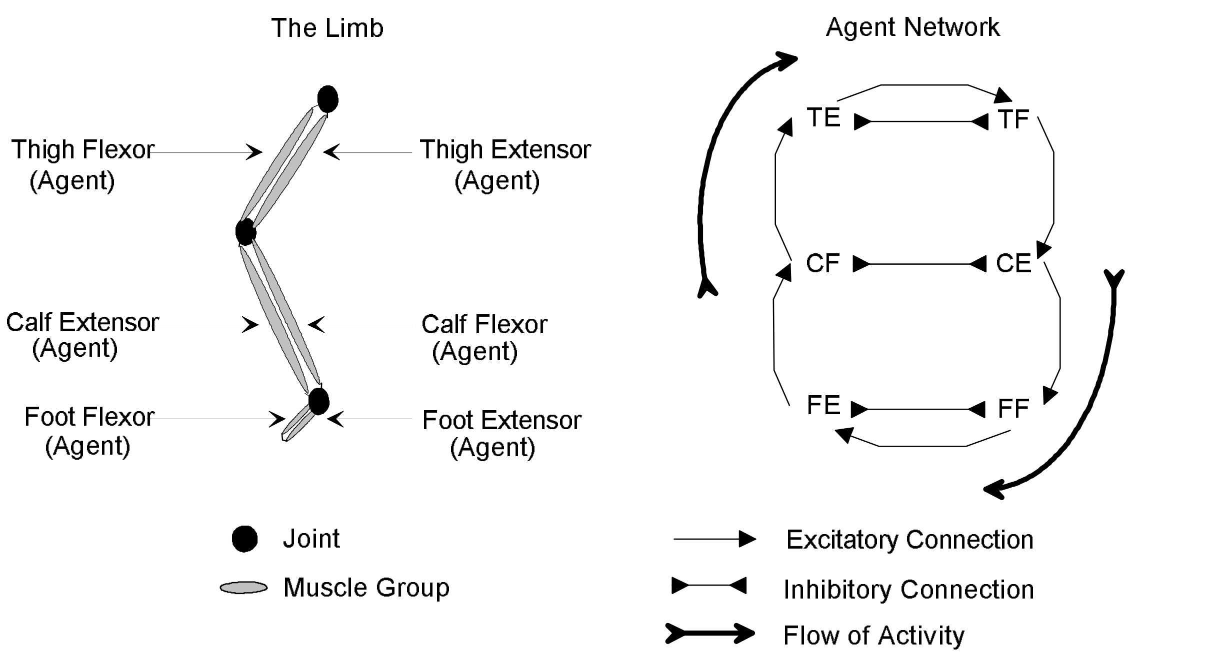
17 See Appendix E Section 2.

18 See Appendix E Section 3.

19 See Appendix E Section 4.

Following a concept first posited by Minsky in his book *Society of Mind* (1986) this paradigm proposes that an intelligent system can be built as a society of functionally separate, but locally interacting agents (or modules) running in parallel with each other - much like, for example, the animals that go to make up coral reefs. As such, the design for an agent's command and control system is viewed as being 'sliced' on the basis of the desired external manifestations of behaviour into individual task achieving, behaviour generating modules. Each individual module has its own specific behaviour generating competence, e.g. extension of a limb section, and complex behaviours are seen to emerge out of the coexistence and local co-operation of these modules. For example, with two modules controlling a section of a limb (one for the section's extension and one for its flexion), a basic walking behaviour can be observed simply from the cyclical pattern of activation and inhibition between these modules and the modules for the other limb sections. Indeed, this is how the majority of locomotory (and other behavioural) action patterns within animals are produced (see Fig. 6.1).

Fig. 6.1 - The Possible Control Network For A Simple Limb



In taking such a stance, sensory information tends to be accessed directly by those modules that require it, removing not only the necessity and ability for any form of central decision making, but also the use of world models built in one module for use by others. Yet while this does not exclude the use of abstract symbolic reasoning within individual modules, such usage does not appear to have been implemented within the literature; Brooks, Kaelbling, and Maes avoiding the issue in anything more than a theoretical sense.

Clearly then this approach to situated activity involves a major shift away from traditional techniques, removing the burdens of centralised decision making, as well as bestowing a number of other benefits upon cohering designs. Improving the system's overall competence is simply a matter of adding new modules, and the distribution of such competence leads to a system which is robust in the face of module failure. Furthermore, at one level of abstraction, the use of behaviour generating modules may share greater similarities with biological systems than more traditional approaches. To summarise, the behaviour-based paradigm is attractive for many reasons, but mainly because it is:

i. Modular.

ii. Distributed.

iii. Flexible.

iv. Robust.

6.4 Problems The Behaviour-Based Approach

Just as the behavioural breakdown of an agent's command and control system carries with it a number of advantages for achieving situated activity, so too does it impose a number of important problems. These being:

i. Problems concerned with efficiency.

If sensory input is dealt with by each module that requires such input, then there will undoubtedly be large repetitions in processing. While it may be argued that even with humans there are a variety of CNS sites where raw perceptual input is processed, with the manner of that processing depending upon its final use, it is also true to say that the results are often then available to whichever mechanisms require them, i.e.

perceptual processing trends not to be duplicated. Take for example visual processing, colour being processed in one area, movement in another, shape in yet another. While the outputs of each processing area do go to specific mechanisms for producing 'low level' behaviours, those outputs are also re-combined in various places and in various combinations for producing 'higher level' behaviours. An agent which exhibited the amount of computational repetition within its command and control system that Brooks (see Appendix E Section 2) appears to be calling for would in all likelihood be one which was unviably inefficient.

ii. Information Fusion.

Generally, as we have seen in Chapter 2, it is likely that within an agent there will exist multiple sources of information, both overlapping, and separate. The fusion of such information is seen as a key component leading towards agent behaviour which is viable given its domain, and this is not seen to occur within systems utilising physically grounded representations.

iii. Module co-ordination problems.

Accompanying the adoption of this design stance is the Module Selection Problem (Maes 1991). Given that an agent is likely to have a number of options open to it for action at any one time, of crucial importance is the co-ordination of individual modules, such that the society as a whole acts in a coherent and viable manner. Thus, adopting the Behaviour-Based approach to the design of an autonomous agent's command and control system requires that any implementation address questions such as:

a. How is it determined if and when a module is active or not?

b. What factors determine module co-operation?

c. Which modules dictate the agent's global beliefs and actions at any one time?

In response to the problems of information fusion, and processing redundancy (outlined above) Kaelbling (1986 & 1991) has proposed a hybrid approach to the design of an agent architecture, as embodied within Situated Automata (see Appendix E Section 2). In keeping with more traditional approaches, at a gross level an agent is viewed as being internally divided on a functional rather than behavioural basis, having a major vertical division between two separate, but interacting automata (finite state machines) - the perception and action components (Kaelbling & Rosenschein 1990). However, in keeping with Nouvelle approaches and the philosophy of hybridisation, each of these two main components is then divided 'horizontally', the perceptual component in to layers of abstraction (from raw sensory input to abstract world models), and the action component in to behaviour producing modules. This approach is definitely a step beyond what Brooks and Agre & Chapman have proposed, allowing an integration of symbolic and non-symbolic processing at different conceptual levels. However, it suffers from a number of problems, not least that the theoretical limitations are not well understood (Wooldridge & Jennings 1994), and the problems associated with module co-ordination which appear to dog all of the behaviour-based approaches to situated activity.

Maes (1989) has shown that the above questions (within Section 6.4.iii), concerning the module selection problem, can be answered via three general methods:

i. Hand code (or hard wire) the control flow, e.g. Brooks (1986), Kaelbling (1989), and Chapman (1990)

ii. Use a meta-level to tell agents when they are active or not.

iii. In rule-based systems let each module's conditions determine whether it acts or not.

iv. Utilise agent network architectures, e.g. Maes (1989).

In contrast to the manner in which Brooks and Kaelbling have implemented behavioural co-ordination, Maes has taken an altogether different route, borrowing ideas from neural networks, to form, what she terms, the *Spreading Activation Network* model (see Appendix E Section 3) for the command and control of intelligent autonomous behaviour. In service of this is an architecture constructed from two types of interconnected competence modules, those concerning actions and those concerning beliefs (although the distinction is not always clear).

Through the links connecting modules activation energy spreads (tuned via the alteration of a number of global system parameters), causing belief or action modules, whose threshold is reached, to fire. The idea is thus both simple and relatively easy to implement, and furthermore it has recently been extended to provide a degree of learning.

6.5 Summary

Traditionally AI has attempted to demonstrate sophisticated reasoning in rather impoverished domains, utilising explicit symbolic representations, and an internal architecture divided on the basis of functionality. However, both of these traits have led to near intractable problems when attempting such reasoning in real-world domains. In juxtaposition with this, Nouvelle AI has taken an entirely different approach to the implementation of situated activity, either doing away with the traditional manner of representing domain information, or the functional decomposition of an agent's architecture, or both. Utilising such approaches Nouvelle AI has tended to concentrate on less sophisticated tasks than traditional AI, but crucially, tasks that operate robustly within complex domains (Brooks 1990).

Agre & Chapman's Pengi System is important because it demonstrates that the use of abstract symbolic representations within a traditionally decomposed architecture is not a necessity. However, this lack of abstractness causes problems in terms of learning and reasoning. Going one step further than this, Brooks' has advocated a behavioural rather than functional decomposition of an agent's architecture, doing away with both the traditional view of agent design and symbolic domain representations. Brooks' approach clearly demonstrates that hierarchical control models (e.g. as posited by Powers - see Appendix A) are both possible, and viable within embedded agent designs. However, the approach does give rise to a number of problems, not least the manner in which information is processed, behaviours co-ordinated, and new information learnt. In response to the first of these problems Kaelbling has proposed the Situated Automata approach, an approach which integrates, to some extent, symbolic and non-symbolic processing. However, as with Brooks' Subsumption Architectures it can be argued that this still suffers from the problems of behavioural co-ordination, and learning. Maes' answer to these has been to take a radically different view of agent design, proposing the use of spreading activation networks rather than behaviour producing modules. Such an approach is of importance because it addresses the problems of behaviour co-ordination within Nouvelle architectures, their lack of learning capabilities, and also their lack of complex reasoning. However, as with the other Nouvelle approaches, I do not believe that this will ever exhibit the full range of reasoning and learning abilities open to human like agents; not because of the modularity involved, but because of the insistence that no explicit representations need be utilised. I am rather tempted towards the idea that a mixture of ideas from both traditional and Nouvelle paradigms will be necessary if a truly autonomous and intelligent embedded agent is to ever be developed, a mixture which must exceed any utilised in the above. The following chapter puts forward a theoretical model for an architecture which does just this, allowing an integration of both planning and learning within a coherent resource sensitive embedded agent.

7. THE AIMAE ARCHITECTURE

In previous chapters we have seen that an agent embedded within a real-world environment is likely to suffer from a whole brigade of constraints; constraints which must be addressed if that agent is to exhibit behaviour which is both intelligent and viable. However, although a number of systems, following a number of approaches, have been posited, none have succeeded in putting forward designs which fully address these constraints. Indeed, classical planning systems, having focused on complex reasoning and/or learning (e.g. TWEAK, NOAH, SOAR, and HACKER. See Section 4), have largely ignored the essential need for reactivity. Conversely, the reverse is true for reactive planning systems (e.g. Universal Planning and Subsumption Architectures. See Section 4). Yet, even with the advent of more 'integrated systems' attempting to mesh reasoning, learning, and reactivity, only limited successes have been enjoyed. Furthermore, problems have arisen regarding the manner in which an agent's domain should be represented, if at all. Utilising abstract symbolic representations elicits deep philosophical questions concerning the nature of the symbolic set to be used, as well as the 'depth' and 'completeness' that an agent's domain should be represented too (see Section 4.1). However, avoiding these problems via the eschewance of symbolic representations, simple brings about a different set of problems (see Section 6.2); as the saying goes, you can run but you cannot hide! For example, how is abstract reasoning, or learning to be achieved if an explicit symbolic set is not utilised? This last point must be emphasised. Without an ability to learn an agent will be continually locked into behaviour which it cannot improve, or alter as domain changes require. It should thus be seen, that along with reactivity and deliberation, learning is one of the most important tasks an agent can, and must, carry out if it is to remain viable. A system exhibiting an effective combination of these has yet to be posited for use within a real-world domain; but why?

It is possible that one reason is that the vast majority of systems have tended to fall, or more accurately, have tended to be pushed into apparently opposing groups, groups whose members exhibit broadly the same types of structure or behaviour. For example, Knowledge-Based vs. Behaviour-Based systems, and Classical vs. Reactive systems. Yet, in the same way that Communism inherently prohibits individual profit making, such artificial divides, adhered to for whatever ideological reason, clearly limit the potential and actual scope of such systems. Thus, in response to the evident need of a system which attempts to cross the divide between reactive, classical, Knowledge-, and Behaviour- based paradigms, I put forward specifications for AIMAE, an **A**gent **I**nformation **M**anagement **A**nd **E**xecution system.

7.1 The Architecture

Having said all of the above, the proposed AIMAE architecture sits clearly in the Knowledge-Based camp when it comes to its internal divisions, being divided into functional information processing modules. This is not because I believe that fully distributed designs, as propounded by those within the Behaviour-Based 'camp', are likely to fail at producing reactive behaviour, or at producing architectures which are robust and easily expandable, but because such designs cannot account adequately for a variety of phenomena which are seen to characterise intelligent behaviour. For example, the fusion of spatially and temporally separated knowledge, the abstract determining of which goals to attempt and in what order, the notion of time, etc. Consequently, I do not see such systems as being capable now, or ever, of scaling up to produce complex cognitive behaviours and accounting for a large number of the requirements that must be met by an *intelligent* autonomous embedded agent. Significantly however, while this may be the case, I do believe that a proportion of the ideas utilised within distributed systems will have an important role to play in AIMAE's various internal modules. As an example, Brooks' ideas concerning subsumption architectures may play a central role in the physical execution and co-ordination of system actions (cf. Torras 1993). Equally, while I see it as likely that an agent's command and control system will be a virtual construct, divided into various functional information processing modules, that virtual construct must also be grounded in something physical. In all higher-order animals alive today, that physical something is a distributed neural network (e.g. Fig. 7.1).

Be that as it may be, it should be noted that while some of a system's internal modules may be implemented using ideas gleaned from the Behaviour-Based paradigm, this does not mean (as Brooks appears to state it does, e.g. 1991a & 1991b) that every architecture we design must be fully implemented *as* a distributed physical system. Rather, it may be enough that such an architecture is implemented simply *on* a distributed system, whether that be real (a transputer network for example), or simulated. In support of this is the idea that the mapping between the real and the virtual may be highly complex, especially where there exists not just one virtual layer, but N virtual layers. Indeed, these layers may be organised such that each influences the others in a manner which need not be strictly hierarchical (cf. Sloman's behaviour production hierarchy, 1992d).

Fig. 7.1 - The Mapping Between Virtual & Physical Architectures

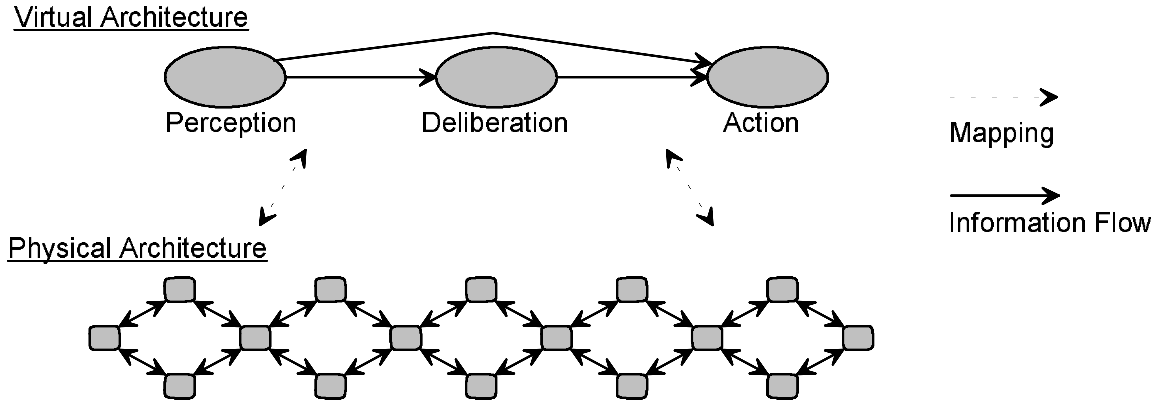
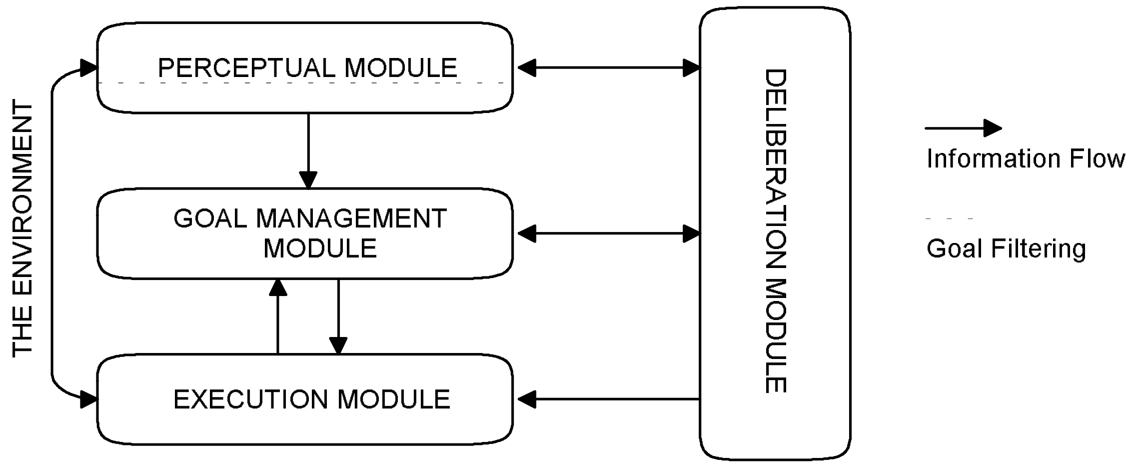


Fig. 7.2 - Schematic View of the Internal Divisions Within AIMAE



Finally, if the structure of an agent's command and control system can be viewed as following both Knowledge and Behaviour Based paradigms then it is highly probably that so too can other areas of that system. For instance, the use of activation levels (as utilised by Maes) for 'action' selection'. However, before detailing the workings of AIMAE's various internal modules, it is worth examining a schematic overview of the system's overall architectural breakdown, and this is shown within Fig. 7.2.

In keeping with the 'broad but shallow' design philosophy being followed while investigating the means by which planning and learning may be integrated, AIMAE's Perceptual, Deliberative, and Execution module are not discussed here, although they are detailed within Appendix F Sections 1, 2 & 3 - respectively.

7.2 Goal Management Module

AIMAE is goal directed (requirement 3.6.i) and hence the Perceptual Module (see Appendix F Section 1) must provide the system with goals in accordance with system concerns and the current environment. These are filtered within the Perceptual module in accordance with requirement 3.3.i, and goals that pass the attention filter (cf. NML1) enter the Goal Management Module. Here a large proportion of an agent's goal scheduling, and plan creation occur, such that resultant behaviour is both goal directed, and reactive. Furthermore, the 'action' selection mechanisms employed ensure that learning[20] is carried out continuously and with respect to the agent's behaviour at any one time. Crucially however, the Goal Management Module does not carry out explicit deliberative processing (this occurs elsewhere within the system), such as look-ahead, predication, or feasibility testing. Hence, it manipulates goals, sub-goals, and the 'methods' by which these may potentially be addressed in a purely heuristic manner, and as such this module can be viewed essentially as a precursor to more advanced behavioural control, possibly in the same way a human brain would operate without full use of the frontal lobes. In many respects therefore, although AIMAE's command and control system is divided into purely functional information processing modules, there still exists the Brooksian (to take just one proponent) concept that behaviour is produced in a hierarchical fashion, with various levels subsuming each other (Fig. 7.3).

7.2.1 Goal Assessment

[20] Within the Goal Management Module the type of learning carried out is reinforcement learning. This is in contrast to inductive or analytical learning, which should be seen generally as requiring computational processes of a type more complex and deliberative than possible within this module.

In keeping with the idea that an agent's behaviour will be constrained by a variety of constraints, the assessment as to whether a particular goal should be *adopted* or *rejected* prior to any 'high level' processing of that goal is vital if the agent's behaviour is to be viable (requirement 3.4.i). An agent that attempts to achieve goals which are either impossible (having passed hard real-time constraints - requirement 3.4.ii), or impractical (as determined partially by soft real-time constraints and utility calculations - requirement 3.4.iii), is likely to exhibit behaviour, which while not necessarily maladaptive, is not optimal either.

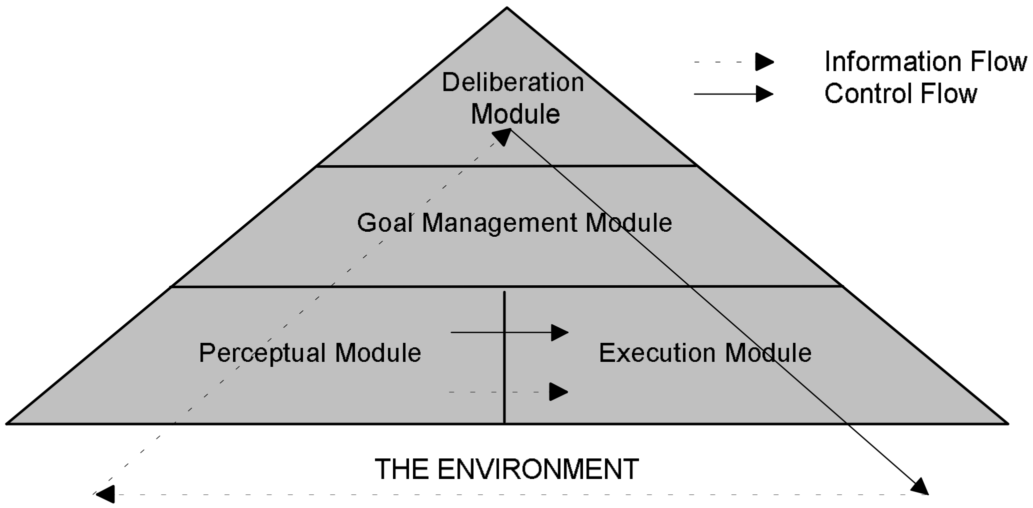
This causes a problem; to make any form of accurate, or near accurate judgement concerning these factors, requires that the agent carry out goal processing[21], where generally, the more processing that is carried out the more accurate the resultant assessment. Yet the whole idea of assessment in the first place, is to reduce the loads placed upon deliberation mechanisms by determining which goals are not worth such deliberation. This underlying concern means that at this stage in a goals 'life', goal assessment must be carried out in a heuristic fashion based upon a limited amount of information which is ready to hand. Indeed, the basic premise can be seen as being equivalent to one of those held by the proponents of Behaviour-Based AI, that is, that the world affords (in Gibsonian terms) an agent with a certain amount of information directly, without that information having to be derived via any high level deliberation processes. For example, a chair affords us information concerning our ability to sit on it, or more subtly, the colour red in nature affords us information concerning possible danger. In an analogous sense, goals can be seen as affording an agent a degree of low level information concerning their suitability for adoption. However, because goals are abstract constructs generated by the agent's command and control system rather than by its environment, the information they afford must either be innate to that agent, or be acquired through learning.

Within AIMAE, associated with goals, are a number of measures, one of which being the average time goal satisfaction requires under various environmental conditions

[21] Indeed, one of the points raised by Beaudoin within his thesis (1994) was that the assessment as to whether goals should be adopted or rejected cannot occur accurately prior to any goal processing. Consequently, it was posited that assessment and goal processing should be seen as occurring cyclically within a system.

(see Section 7.2.6), and another the measure of the utility that such goal satisfaction can give rise to[22]. AIMAE's goal assessment mechanisms are aimed at being able to utilise these measures in conjunction with domain knowledge to provide quickly computable assessments as to whether a goal should be adopted or rejected. However, in keeping with requirement 3.4.iv, where information extra to, or information more accurate to, that which is known, is required, it is proposed that the mechanisms can always request help from the Deliberative Module (see Appendix F Section 2).

Fig. 7.3 - Hierarchical Control Within AIMAE



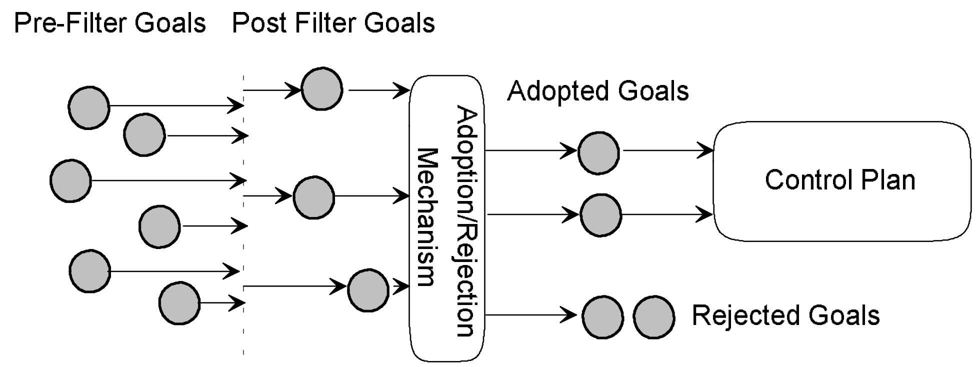
Let me give an example. The goal to go to the bank has an associated time measure of approximately 15 minutes, and an associated utility function that shows that utility decreases markedly after a soft-time constraint of less than approximately 10 minutes, i.e. if you have at least 15 minutes to go the bank you are ok, but anything less than 10 makes it pointless. In conjunction with known information concerning the fact that you have a meeting in about 20 minutes, the goal can be adopted. However, if the information concerning the current time (in this example) was not known, or known to be unreliable, a request for help could be sent to the Deliberation Module. This module

22 However, because utility tends to alter with the time spent achieving a goal, as well as the conditions under which that achievement takes place, this utility measure tends to be an function rather than a single number (see Section 4.2).

(as detailed within Appendix F Section 2) is able to generate new goals or actions directly, and in this case might generate the direct actions of looking at a clock.

Adopted goals are forwarded into the control plan mechanism which carries out the tasks of heuristic goal expansion[23] and scheduling, while rejected goals are effectively 'deleted' from the system (Fig. 7.4). However, it should be noted that while the assessments carried out so far have been heuristic, and thus only partially accurate, the system does have the ability to carry out more accurate assessments once goal satisfaction has started to take place, i.e. once a goal has been partially expanded in to a plan and more information concerning that goal and plan is available. This form of 'active' assessment is seen as being carried out by the Deliberation Module in conjunction with the control plan, and gives rise to measures that more accurately reflect the goal's importance[24], urgency, and utility. These are combined to form a measure similar to insistence, termed *Intensity*.

Fig. 7.4 - Goal Flow



7.2.2 Goals And The Control Plan

Once goals have been adopted it is imperative that the system organise those goals such that their potential achievement is attempted with maximal utility, i.e. adopted

---

23 Goal Expansion is the term used to describe the task of *expanding* a goal in to the executable actions which may solve it, i.e. planning.

24 As opposed to the use of importance within the Insistence measure. Here importance should be seen as being not only a measure of a goal's dispositional congruence (see footnote m - In Appendix E), but also its congruence with other already extant system goals.

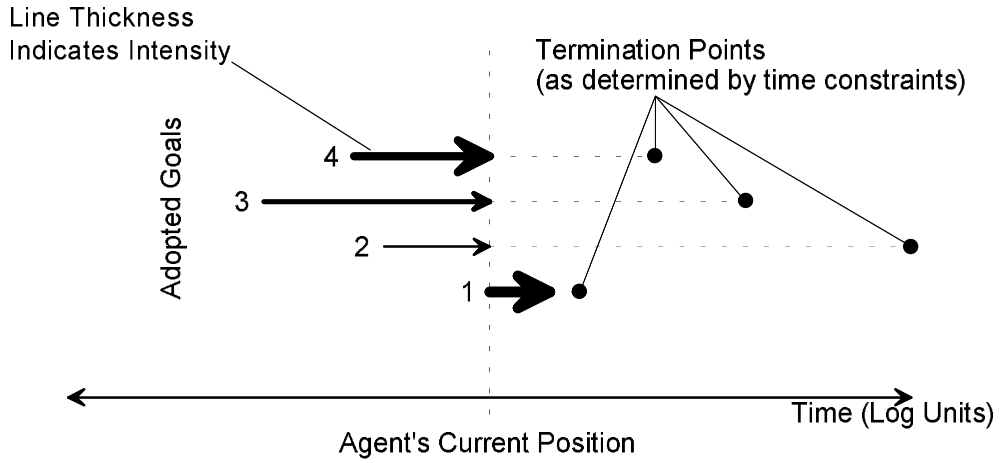
goals must be placed within some form of schedule (requirement 3.5.i). Furthermore, such a schedule should not be fixed - new goals may be adopted which require 'attention' prior to older goals, or environmental changes may dictate ordering changes. Consequently, in accordance with requirement 3.5.ii, the scheduling system utilised within an agent's command and control system must be dynamically re-orderable. This is achieved within AIMAE via the use of a *Control Plan* system, allowing concurrent multiple goals to be active at any one time, and supporting a notion of time. The control plan takes the form of a time-graph structure, extending on the x-axis away from the agent's present point in time towards both future and past points, and extending on the y-axis to encompass all of the agent's currently adopted goals (Fig. 7.5). In this figure it can be seen that for each of these goals there extends forwards from the agent's current point in time 'time lines' terminating at a variety of points. These termination points represent points in time beyond which goal achievement is either not possible, or not beneficial in terms of potential utility. Termination points are not fixed, but remain where placed as determined by the heuristics within goal assessment mechanisms. However, as mentioned above, once a goal has started to become partially expanded its termination point may be recomputed more systematically within the Deliberative Module, i.e. termination points are heuristically determined on the basis of goal information, or deliberatively on the basis of goal utility[25]. Furthermore, some termination points may extend so far forward that they do not need to be, and therefore are not, represented within the control plan.

Termination points are useful because they play an important role in influencing each individual goal's assigned intensity value (indicated in Fig. 7.5 by the thickness of each time line). The nearer the agent's current point in time to a termination point, the

[25] I say utility because deliberative 're-positioning' of Termination Points should not simply take into account extra real-time constraint information, but also the associated costs and benefits of achieving goals with respect to those constraints, i.e. the potential utility of goal achievement. Take as an example the goal to gain a PhD. A hard constraint is that the thesis must be submitted within 10 years of the PhD being started, and a soft one that funding only lasts for 3-4 years. However, the calculated termination point for finishing should not just take into account these constraints, but also the utility of obtaining a PhD. If it is likely to be very useful then it will be viable to position the termination point after the soft time constraint, i.e. after funding has run out. Conversely, if the PhD is less useful then an agent must decide that the hardships that would be encountered attempt to complete without funding do no warrant the potential benefits. Thus, the positioning of termination points may be highly complex, involving numerous factors, and consequently non-heuristic positioning is seen both as being a Deliberative Module task, and one which may be re-addressed any number of times as new information becomes available.

greater the intensity value assigned to the particular goal in question (as determined by the utility function associated with a particular goal). Although I have not covered the following part of the control plan mechanism yet, the intensity value of a goal plays a large part in the selection and activation of actions which are aimed at satisfying that goal. Thus, increases in a goal's intensity value on the basis of its approaching termination point, consequently leads to increases in that goal's likelihood of being made active prior to that termination point, i.e. it leads to an increase in an agent's ability for timely responses (requirement 3.6.iv).

Fig. 7.5 - Control Plan Structure

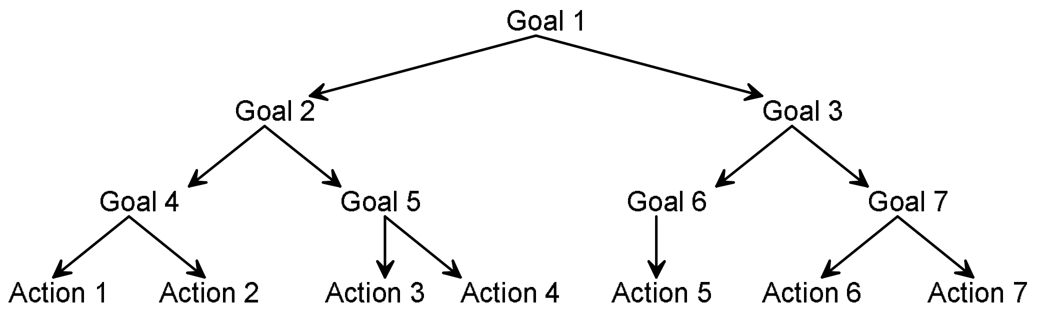


It should be noted, however, that the termination point associated increases in intensity values are above and beyond any intensity value increases that are imparted by generator re-activation. This dual approach to intensity increases, therefore, not only allows for timely responses, but responses which are very dependent upon the environment. Let me give an example. If you are hungry you may generate a top level goal to eat, and from this a goal to buy sandwiches. The placement of this latter goal within the control plan will also give rise to a termination point for satisfying that goal, possibly the closing time of the sandwich shop. As this point is neared the goal to buy sandwiches will be made more intense, indicating that you are nearing a point beyond which you will not be able to obtain sandwiches. However, in addition to this, the motive generator may also have been re-triggered, indicating that you are still hungry (or even getting hungrier). Thus, the intensity value assigned to the goal to buy sandwiches increases at a rate faster than if only one mechanism were used to increase it.

7.2.3 Goals & Strategies

Classically, the process of plan creation is seen to utilise techniques such as search, back tracking, and sub-goaling, etc., where any resultant plans can be viewed as forming multi-layered 'AND' tree hierarchies which move the agent from its current situation towards that of the goal satisfying situation (requirement 3.6.iii). Such hierarchies originating at a top-level goal, traversing a variable number of sub-goal branches, and terminating each of these branches with actions (Fig. 7.6). Furthermore, all of this is seen to occur prior to any of the resultant actions being executed.

Fig. 7.6 - A Plan Tree



Utilising such a plan construct, an agent's command and control system simply has to execute each of the actions (generally in their specified order) to bring about the hoped for goal satisfaction. Yet we have seen that this view of planning is not a particularly viable option within real-world environments, i.e. ones where the actions required to bring about goal satisfaction may change from moment to moment. Hence, planning must be achieved such that changing environmental conditions have an influence over the developing plan (requirement 3.6.v). However, although it is likely that even the earliest researchers realised this essential need for context dependent planning within real-world environments, it has not been until quite recently that the requisite computational resources have been made available for such integration to become possible. Consequently, much of the earliest work on planning neglected the problems associated with real-world behaviour either by simplifying certain characteristics of such a world (e.g. classical planning - Section 4.2), or by reducing the planning carried out within it (e.g. reactive planning - Section 4.4). Yet by ignoring one aspect of real-world command and control in favour of another, much of what was discovered was of limited use in such worlds, and typical of this are the problems associated with real-world planning.

As mentioned previously, the earliest 'classical' style planners tended to solve goals via complete search through the goal's associated problem space. However, whilst this was viable within simple environments, it became increasing intractable as environmental complexity grew, and planning became exponential. Furthermore, all the domains initially chosen were static environments where there existed no limits on the time available for such problem solving. As this was addressed the time available for problem solving became constrained. Thus, in complex and dynamic worlds, planning systems are faced with a dilemma; if problem solving is to show completeness then the whole problem space must be searched, however, this requires time which is generally not available. Hence, for planning to occur viably within real-world domains, various methods had to be developed to reduce the computational burdens placed upon the planning system, e.g. the use of sub-goals and abstraction hierarchies, etc. (see Appendix B). Of prime importance was the concept underlying the development of Case-Based planners, i.e. that planners could utilise stored sub-plans in the solutions to new problems and hence save on problem solving search. Similar to macro-operators such sub-plans specify a small section of the required plan, rather than the plan in its entirety, and are advantageous for a number of reasons, three of which being:

a. Much like Minton's S-Macro's (1985), sub-plans provide the system with a method of forming plans to satisfy any goal without duplication in the actual stored information. Hence, the use of storage facilities is highly efficient. This only becomes fully apparent when there exists multiple goals for which plans must be stored, and/or when there exist multiple plans for any of these goals.

b. Improving one sub-plan (in terms of speed or efficiency, etc.) means that all the plans that utilise it will also improve without them all having to be explicitly altered.

c. Using a sub-plan leads to a reduction in the amount of search that is required to satisfy a goal.

Fig. 7.7 - The Strategies For The Plan in Fig. 7.6

Goals	Strategies
Goal 1 =	Goal 2 + Goal 3
Goal 2 =	Goal 4 + Goal 5
Goal 3 =	Goal 6 + Goal 7
Goal 4 =	Action 1 + Action 2
Goal 5 =	Action 3 + Action 4
Goal 6 =	Action 5
Goal 7 =	Action 6 + Action 7

For these reasons (the need for reactivity and deliberation, as well as increased search efficiency) AIMAE does not explicitly represent knowledge concerning how to achieve various goals as complete plans, but rather as specialised sub-plan constructs called *strategies*. These are much simpler than multi-layered plan trees, being single layered structures corresponding to a particular goal along with the directly linked sub-goals and actions that branch from it (Fig. 7.7). Because each strategy is really a sub-section of an overall goal satisfying plan, each one takes the AND form, i.e. all of a strategy's components must be executed if that strategy is to succeed (potentially). However, this is not to say that the ordering is fixed between those components. In a strategy to make coffee taking the form of {get cup - boil water - make drink}, the first two steps can conceivably be swapped without effecting the success or failure of the last step. Indeed, step two involves the initiation of a *process*; an event which once started continues for a period of time of its own accord, i.e. the boiling of water (see footnote 26). Thus, once this is initiated, getting the cup may be effectively carried out in parallel with it (see Section 7.2.9). Furthermore, getting the cup while the water boils will effect the strategy's utility, increasing the efficiency of time usage (for want of a better term). Thus, although strategies do take the AND form, they may also include flags specifying certain parameters, such as whether their internal ordering is strict, i.e.

whether they are linear, fully non-linear, or partially non-linear. As we shall see within Section 7.2.9, where such orderings are not strict, it is usually up to the control plan mechanisms to determine what order strategy components are carried out, and possibly interleaved.

In addition to what might be termed a strategy's meta-level grammatical structure, is the grammatical structure that each component itself takes. Generally, a component will specify simply an action, or a goal, however, it is not infeasible to expect to see as one of a strategy's components a loop function of the form, DO X UNTIL CONDITION Y, e.g. tighten screw until tight. This is termed a *cycle*[26]. However, although a component may take a non-simple form, such as a loop structure, it cannot specify predicates of the IF... THEN form, as this represents a departure away from the AND nature of strategies; such choices being seen to occur within entries of a goal look-up table (see Section 7.2.5).

Thus, in attempting to execute actions which will hopefully lead to the satisfaction of a top-level goal, AIMAE will utilise a number of strategies, ranging from those applying to the top-level goal, to those applying to goals much further down the 'plan tree'. This is thus similar to the manner in which Case-Based planners re-use plan 'chunks', e.g. CHEF (see Appendix D Section 1). However, unlike these other systems, the manner in which AIMAE utilises strategies in the satisfaction of goals gives rise not only to the advantages listed above, but also to further advantages in terms of flexibility of plan creation, integrating reaction and deliberation, and certain learning abilities. In summary a strategy is as shown in Fig. 7.8.

Finally, strategies, actions, and processes presuppose a large number of facts concerning the state of the agent's environment prior to, and during their execution. Thus, as with all planning systems, the usage of strategies and physical behaviours must be

[26] It should be noted that the form of a loop structure is different from that of a process. A process is an event or sequence of events which once initiated extend over a period of time without need of intervention by the agent. As an example, the process of filling a cup once started will continue so long as the cup is under the flow of water; the agent is not required to keep turning the tap for instance. Consequently, although the action(s) required to initiate a process are generally standard system actions, because they result in processes, they themselves are termed process initiators. Converse to this are sequences of agent actions which occur in a cyclical manner. For example, tightening a screw, hammering a nail, walking. Such manifest behaviours do require continual involvement by the agent, and as such they are markedly different from processes (and process initiators), and are termed cycles.

accompanied by a set of invocational pre-conditions as well as a set of existence conditions.

Fig. 7.8 - A Strategy

Component List

HEADER: [(component 1) + (component 2) + ... + (component x)]

(Operator) (Order Parameter)

Goal / Action / Cycle / Process Initiator

True / False

Goal / Action /. Process Initiator

7.2.4 Behavioural Conditions

In making decisions concerning the achievement of goals an agent is effectively delimiting the possible courses of behaviour that it can follow, whether that be at an abstract level, e.g. via strategies and goals, or at a more concrete level, e.g. via actions. Furthermore, the 'closer' decisions are to determining actual physical behaviour, the greater the commitment of an agent to expending resources. Consequently, if maladaptive behaviour is not to result, such decisions should be governed by a number of factors, not least the goal in question, and the environmental conditions at any particular moment in time. All behavioural operators within AIMAE may be associated with pre- and extended conditions[27], and this is as true for goals and strategies, as it is for processes, cycles and actions. For real-worlds where behavioural operators may have a large number of interacting conditions, deliberation as to which conditions a system should attempt to meet, or re-establish, if at all, must be seen as decidedly non trivial, and is carried out within the AIMAE system via the Deliberation Module discussed within Appendix F Section 2.3.e.

[27] See Appendix F Section 4.

7.2.5 Using Strategies To Form Plans

Within Section 7.2.3 it was shown that goals, and the sub-plans (strategies) that can potentially satisfy them, take the form of explicit symbolic representations within AI-MAE. Below it is detailed how these two constructs interact in the production of agent behaviour.

The task of turning strategies into plans can occur only if those strategies are indexed with respect to the goals that they satisfy, and such indexing is seen to occur within the *Goal Lookup Table*. Here goal entries are associated with pointers to potentially satisfying strategies held within the *Strategy Table*. For example, in Fig. 7.7 goal 1 is potentially achieved via the strategy {goal 2 + goal 3}, let us call this strategy s1. This being the case, goal 1's entry within the Goal Table would have associated with it a pointer for strategy s1, whose details are held within the Strategy Table. Hence, expanding a goal down into its various sub-goals, process initiators, cycles, and actions is a task of repeated cross indexing between these two tables (Fig. 7.9).

The big problem with this, and other forms of plan creation via sub-plan retrieval, is that as an index table grows, search through that index can become more time consuming than search through the original problem space (Minton 1985). In effect nullifying any efficiency gains brought about via the use of sub-plans in the first place. This indexing problem is addressed via the use of ratings (see Section 7.2.6 below); heuristic measures used in guiding plan construction via the provision of a multi-dimensional ordering of sub-plans.

Fig. 7.9 - Cross Indexing Look-up Tables

Goal Table	Strategy Table
Goal 1 = S1	S1 = Goal 2 + Goal 3
Goal 2 = S2	S2 = Goal 4 + Goal 5
Goal 3 = S3	S3 = Goal 6 + Goal 7
Goal 4 = S4	S4 = Action 1 + Action 2
Goal 5 = S5	S5 = Action 3 + Action 4
Goal 6 = S6	S6 = Action 5
Goal 7 = S7	S7 = Action 6 + Action 7

7.2.6 Strategies & Their Ratings

What makes AIMAE's method of representing goals and strategies so flexible, is not the fact that the plan's are stored as 'chunks', this has been done before, but in the manner that such chunks are indexed. We have seen (see Section 7.2.3 above) that strategies take at the meta-level the AND form; every action, loop, or goal specified being necessary if that strategy is to terminate successfully.

However, there are many ways of achieving the same goal, most of which are context dependent, and consequently planning is not just a simple AND process, it also involves choices, i.e. the OR operator is also required. This process is seen to occur within the Goal Table, where entries are not limited to one associated strategy, but rather may be associated with many potentially goal satisfying strategies via the use of multiple pointers, each separated by an OR operator (Fig. 7.10). In this figure, when AIMAE is required to address goal 1, it may do so either with strategy 1, or strategy 3, or any other associated strategy. Let me give a more concrete example... To boil water for a cup of coffee, one can use a kettle, or a saucepan, or even a microwave, and the choice concerning which strategy to use will depend upon (among other things) the agent's current situation. Thus, the use of Goal and Strategy tables within AIMAE constitutes a method of compressing traditional AND-OR plan trees, especially where multiple trees share common sub-sections, and involve loops.

Fig. 7.10 - Multiple Strategies May Solve The Same Goal

Goal 1 = S1 <u>OR</u> S3 <u>OR</u> ...
Goal 2 = S2
Goal 3 = S3 <u>OR</u> S1
Goal 4 = S4 <u>OR</u> S2 <u>OR</u> ...
Goal 5 = S6 <u>OR</u> S10
Goal 6 = S5 <u>OR</u> S2 <u>OR</u> S30
Etc.

As is shown within Fig. 7.10 above, an important point to note with concern to goals and their associated strategies is the fact that a strategy is not limited to solving one particular goal. Much like a saucepan can be used to boil water for coffee, so too can it be used to boil water for cooking vegetables. Due to this potential for strategies to be associated with more than one goal, it should be seen that the competence of the system increases exponentially as new strategies are created and stored. However, as the list of pointers associated with a goal grows, so too does the problem of deciding which strategy should be used in satisfying it. This is addressed via the use of a set of *ratings* assigned to each individual goal-strategy pairing (Fig. 7.11 below).

Fig. 7.11 - Goals, Strategies & Ratings

Goal	Associated Strategies
g1	s1 (ratings g1s1) OR s4 (ratings g1s4) OR ...
g2	s3 (ratings g2s3) OR s4 (ratings g2s4) OR ...
etc...	

AIMAE is able to determine which of the possible strategies to use in an attempt to satisfy a goal via the use of a set of ratings which are assigned to each pointer associated with each goal. This set of ratings forms a specific profile of that particular pairing over a number of dimensions, and is independent of the rating sets assigned to other goal-strategy pairings. This last point must be emphasised. A goal may be associated with two strategies, each having different rating sets. Furthermore, each of those strategies may be associated with other goals, and the rating sets used will also be different. For example, using a strategy which involves boiling water in a saucepan to make coffee should clearly have a different performance profile to using a strategy that involves using that saucepan to cook food.

Clearly, a whole variety of both qualitative and quantitative measures could feasibly be utilised within a system, e.g. safety, aesthetic value, moral integrity, etc. Indeed, it is likely that the ratings used will depend upon the strategies in question and the types of goals that they are aimed at satisfying. For instance, the strategies used in producing gymnastic behaviour may have markedly different rating sets to those used in car fixing, and both sets may be present within the same system. However in addition to this, account should be made not only of the particular goal-strategy pair, but also of the context in which that pair occur. For example, using a saucepan to boil water is likely to have different ratings when using a camping stove than when using a kitchen cooker. Although it is possible that such differences in context could be differentiated within the pre-conditions assigned to strategies, this would result in multiple versions of effectively the same strategy, one for each type of context in which that strategy might be applied. This is obviously not a very efficient way of dealing with context dependent effects on strategy performance. Hence, it may be advantageous and necessary if the concept of non-stereotypical rating sets is extended to allow multiple sets to be assigned to each pairing. Such multiple sets could then allow for context dependent influences on strategy performance to be taken in to account without the need to duplicate the pairing in question. Using such an extension of the rating concept would require that the Deliberation Module (see Appendix F Section 2.3.f) choose not only which rating type(s) should be used by the Control Plan in strategy selection, but also which rating sets.

Finally, it should be emphasised that there is no reason why rating types must be of one metric form, i.e. qualitative or quantitative. For instance, the time rating, could use qualitative values such as, long, very quick, etc., just as easily as numerical ones. Furthermore, ratings could as easily be a product of complex computations carried out deliberatively, as simple ones carried out heuristically, or the even the results of various neural networks. The potentials are enormous, and limited only by a system designer's ingenuity and need. However, for the purpose of this paper - the investigation as to whether ratings have a role to play in the integration of planning and learning - I have chosen four heuristically determined numerical rating types for use within the AIMAE system. These being:

a. Average degree of strategy execution *satisfaction*: range 0 - 100 %.

A representation of the probability that the strategy in question will satisfy the goal in question[28]. When a strategy succeeds at run-time it is assigned a value of 100%, conversely, when it fails it is assigned a value of 0%. This saves the system from having to carry out the computationally difficult task of determining *how* satisfying a strategy was; it either worked or it didn't. However, although a strategies run-time measure will be either 0% or 100%, its stored satisfaction

[28] Strictly speaking, the majority of ratings (except the frequency rating) are a representation of past goal-strategy performance, not a predictive calculation of future performance. However, it is as an indicator of such future performance that they are interpreted and used by the system.

measure may be any value between these two. This is due to the algorithms used to percolate ratings 'between' strategies and then combine them with the 'old' stored values to produce 'new' values for storage (hence the use of the term *average* within the heading). This 'calculation' of percolated ratings is further discussed within the next section (7.2.7).

b. Average degree of strategy execution ***efficiency***: range 0 - 100%.

A representation of the costs vs. the benefits of a strategy[25]. While the mechanisms for the measurement of efficiency are unclear at this point in AIMAE's design, there is no doubt that animals do have, and do use efficiency measures in determining behaviour (e.g. Krebs & Davies 1987). Again this is an average measure indicating that run-time results are used to update stored results.

c. Average *time* required for strategy execution: range 0+.

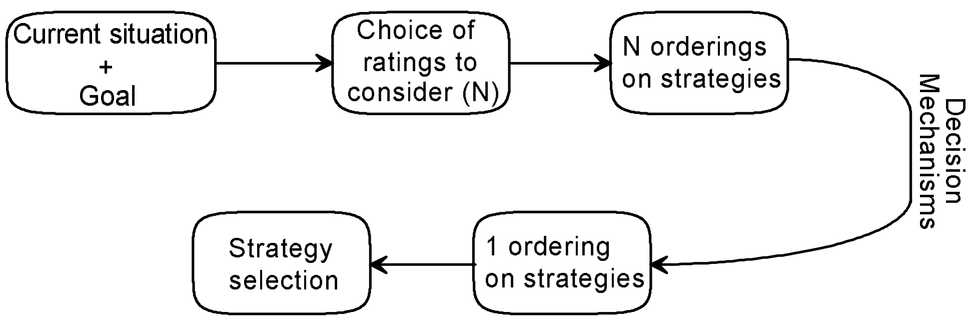
A measure of the time required for the strategy to execute through to termination[25]. As with the previous two ratings, this is calculated at run-time and does effect the stored value assigned to a particular goal-strategy pairing.

d. *Frequency* of strategy use: range 0+.

This is a measure of how often the goal-strategy pairing has been used, and is increased every time the pairing is used regardless of whether the strategy is successful or not. This is important. Adjusting the frequency rating only when the strategy has succeeded gives rise to an obvious problem. The system can only determine whether a strategy with a low success value is really bad if it knows how often the strategy has been attempted. To elaborate on this, because of the updating of stored ratings, the more often a strategy is called the more accurately should its associated ratings represent its performance. Thus, a strategy that has been called five times and has a rating of 30% is probably worse than one which has been called five times but has a rating of 50%. However, this may not be the case if the first strategy had only been called once, implying that its ratings may not be as representative of its performance as those of the second strategy. Thus, without an accurate frequency rating assigned to each goal-strategy pairing comparisons during deliberation may not be wholly accurate.

In Summary therefore, ratings store (dynamically updateable - see Section 7.2.7 below) multi-dimensional orderings on potentially goal satisfying strategies. These facilitate strategy selection based upon the goal in question and the agent's current situation (both environmental and cognitive). Such an overall context determines which subset of orderings will be considered; and decision mechanisms then combine these orderings to form a single ordering that is used in strategy selection (see Fig. 7.12).

Fig. 7.12 - The Use of Ratings In Strategy Selection



7.2.7 Ratings And Learning

The use of strategies in the formation of a potentially goal satisfying plan gives rise to a structure called a *strategy frame*; in effect a plan tree much like that in Fig. 7.6. However, this strategy frame is not formed in its entirety prior to the execution of any of the actions that it specifies. As each strategy is chosen by Control Plan mechanisms (see Section 7.2.9) each component within that strategy must be addressed (via the execution of actions and processes, or the satisfaction of sub-goals) before that strategy is deemed to be satisfied[29]. However, as was noted within Section 7.2.3 above, the order

[29] Obviously if a component fails to terminate successfully then the strategy in question may also fail to terminate successfully. For this reason failed components must be noticed and dealt with in a manner which is adaptive for the agent, and this is detailed further within Section 7.2.9 and Appendix F Section 4.

with which those components are addressed need not be fixed, and furthermore, because some components may be process initiators, or cycles the execution of certain components may be interleaved[30]. Let me clarify this with a simple example of a strategy consisting of three components; a sub-goal, a process initiator, and an action. The sub-goal and process initiator are 'flagged' as not being fixed in order, but the action is 'flagged' as coming after both of these. In a case like this, it is likely that the system would initiate the process and then start to satisfy the sub-goal. However, it should be remembered that achievement of the sub-goal will itself require a strategy, and this may in turn specify certain actions, cycles, process initiators, and sub-goals. Thus, as each component within a strategy is addressed by the system, it leads to the creation of a strategy frame in a depth first manner. Furthermore, because actions, cycles, and process initiators are seen as being the lowest points in a strategy frame (the frame cannot extend below them unlike for sub-goals), goal satisfaction can in some respects be seen as occurring in a bottom up fashion. Actions, cycles, and process initiators being executed to satisfy sub-goals, whose satisfaction partially satisfies higher lever sub-goals, all the way up until the top-level goal is achieved (or not). This has important consequences for the ratings assigned to each goal-strategy pairing.

As was specified within Section 7.2.6 above, when a strategy is addressed by the system a variety of ratings are calculated for it. Except for the frequency rating, these ratings can only be determined when the whole strategy has been addressed, for example, the time required for the strategy to terminate. Furthermore, as posited within Section 7.2.3, each component must be achieved prior to the next component within the strategy being addressed (except for process initiators and cycles), and these components will themselves specify strategies whose components must be achieved, and so on down until system executable actions are specified. The important point being, that such a reliance on component termination prior to rating assessment gives rise to an effect whereby the ratings for one strategy effectively percolate up the

[30] Clearly an agent that has more than one type of output device should be capable of interleaving the use of those output devices in a manner which is beneficial to that agent, for instance, walking and talking. However, in many cases this is not possible, for instance trying to chew on food before it has been put in the mouth (note that the acts of eating and chewing are both behavioural cycles!). For this reason, although the majority of behavioural initiators must be carried out 'serially', I believe that a system should be capable of executing behavioural cycles in parallel with other system actions, so long as those other system actions do not require the same output device as the cycle.

strategy frame influencing the ratings for all of the strategies above it, to the highest level goal, and this is likely to have important behavioural consequences.

Allow me to elaborate. When the run-time ratings of all the components linked to form one strategy have been determined, then the corresponding ratings for each component are merged to give the strategy its ratings. However, this is not carried out in a uniform manner. An average value is calculated for the satisfaction and efficiency ratings, while the time ratings are summed, and if other ratings are utilised they may be combined in a manner which is different again, i.e. the combining of individual strategy component ratings to give overall strategy ratings is not uniform or stereotyped, but varies with each rating type. Furthermore, it may turn out to be the case that some components within a strategy should play a larger part in determining that strategy's ratings than other components, and this may depend not only upon the components and the goal in question, but also upon an agent's dispositions[31]. Finally, when the ratings for a strategy have been determined they are merged with that goal-strategy pair's ratings to form new stored ratings for that pair. For example, given the following Goal & Strategy Table then the calculation of ratings proceeds as described:

<u>Goal Table</u>

Goal 1	=	S1 (efficiency **50**: satisfaction **50**: time **100**)
Goal 2	=	S2 (efficiency **60**: satisfaction **60**: time **20**)

<u>Strategy Table</u>

Strategy 1	=	Goal 2: Action 1: Action 2
Strategy 2	=	Action 3: Action 4

The achievement of goal 1 specifies that strategy 1 be used. This in-turn specifies goal 2 plus two actions. To achieve goal 2, strategy 2 is required, and this also involves two actions. Hence to actually achieve goal 1, the two actions of strategy 2 are executed in service of goal 2. Then, having achieved goal 2 the

[31] It should be noted that the goals an agent forms are not wholly determined by environmental or 'cognitive' events, but also by the goal generators extant within the system. It should be seen that certain high level system concerns, derived either innately or socially, play an important part in determining the generators extant within the system, and as such, the goals possible. Furthermore, such high level 'concerns' are also likely to influence the 'importance' an agent attaches to such goals, not only in terms of their potential benefits, but also their potential costs. The Cognition and Affect project has broadly labelled such 'concerns' *dispositions* (e.g. Sloman & Humphreys 1992).

rest of strategy 1 can be executed to achieve Goal 1, i.e. goal achievement occurs in reverse. The calculation of ratings thus occurs as below, 1st for strategy 2, and then for strategy 1 (ratings for actions have been arbitrarily chosen):

Strategy 2:

(a) Determine run-time ratings for <u>each</u> component of strategy 2:

Action 3 ratings	= efficiency 70 - satisfaction 100 - time 20
Action 4 ratings	= efficiency 50 - satisfaction 100 - time 30

(b) Calculate run-time ratings for <u>all</u> components of strategy 2, using figures in (a):

efficiency	= (70+50)/2	= 60
satisfaction	= (100+100)/2	= 100
time	= 20+30	= 50

(c) Update *stored* ratings for strategy 2, using figures from (b) and **old stored** figures:

efficiency	= (60+**60**)/2	= *60*
satisfaction	= (100+**60**)/2	= *80*
time	= (50+**20**)/2	= *35*

Strategy 1:

(d) Determine run-time ratings for <u>each</u> component of strategy 1:

Goal 1 ratings (from b)	= efficiency 60 - satisfaction 100 - time 50
Action 1 ratings	= efficiency 80 - satisfaction 100 - time 40
Action 2 ratings	= efficiency 70 - satisfaction 100 - time 50

(e) Calculate run-time ratings for *all* components of strategy 1, using figures in (d):

efficiency	= (60+80+70)/3	= 70
satisfaction	= (100+100+100)/3	= 100
time	= 50+40+50	= 140

(f) Update *stored* ratings for strategy 1, using figures from (e) and **old stored** figures:

efficiency	= (70+**50**)/2	= *60*
satisfaction	= (100+**50**)/2	= *75.5*
time	= (140+**100**)/2	= *120*

<u>*Updated* Goal Table</u>

Using figures in *italic* from (c) and (f):

Goal 1	=	S1 (efficiency *60*: satisfaction *75.5*: time *120*)
Goal 2	=	S2 (efficiency *60*: satisfaction *80*: time *35*)

The above example while quite hard to follow is much simplified from what might occur within a system. Indeed, rather than a simple averaging of efficiency and satisfaction ratings in steps (b) and (e), the merger could conceivably be weighted in accordance with the relative 'importance' each component played within the strategy. Furthermore, the updating of stored values via their merger with run-time values, may be weighted more in favour of those stored values, protecting learnt information from 'noisy' behaviour. Indeed, if too little emphasis is placed on the stored-ratings then they may be susceptible to aberrant strategy performances, e.g. ones which are exceptionally good or bad. However, if not enough weight is given to run-time ratings the system is likely to be slow at tailoring its responses to environmental conditions. Thus, not only are there many different ratings that a system could use to profile a strategy, but also many ways the run-time ratings can be calculated, and many ways those run-time ratings can be used to update stored ratings. Indeed, although I do not claim that the above is at all representative of the way that humans store, select between, and assess 'plans', it could be that the determining of these three factors (or more) within an similar system is one factor that plays a large role in the lengthy time course required for human cognitive development, and even adult learning[32].

To summarise, we can see from the above example that the run-time ratings of a sub-goal will influence the run-time ratings of the strategy from which it was called, and hence ratings in some respects percolate up a strategy frame, from the lowest level sub-goals to the top level goal. Furthermore, although the satisfaction rating is an all or nothing value at run-time, its stored value can be any value up to 100%. This use of ratings is vital to a system's behavioural viability (requirement 3.6.ii), allowing it to decide between the various strategies associated with a goal via the use of a number of performance measures. For instance, a system may require that goals be dealt with quickly, and thus speed (time of execution) may be more important than efficiency of execution when choosing between strategies. Such decisions are seen to be made within the Deliberative module and effect intimately the Control Plan mechanisms, as described within Appendix F Section 2.

---

[32] Shing (1994) is looking at how the number of inputs an agent has effects the complexity and thus tractability of learning.

7.2.8 Expanding Goals

It has been shown in the above section how the ratings assigned to each goal-strategy pairing are calculated, but as of yet, little mention has been made concerning the manner in which they (the ratings) are utilised for strategy selection. It is posited that in a manner similar to the use of activation levels within Maes's agent networks, AIMAE's Control Plan mechanisms will utilise ratings in the heuristic selection of strategies (and thus actions) where possible. However, two major differences exist between the two systems:

a. Activation Levels & Thresholds.

Within Maes's activation networks behaviours are activated in a simplistic fashion via the spread of activation energy, modules becoming activated when their energy surpasses a system set threshold. Converse to this is the utilisation of ratings by AIMAE. Firstly, there is no concept of activation energy 'spreading' between behaviour producing modules (strategies)[33], and secondly a strategy will be activated if it has the highest chosen rating amongst the set of possibles, not if the chosen rating is above some threshold (this is not quite correct as will be shown below).

b. Parameters that may alter Selection.

As touched upon in (a) above, the threshold level for activation of modules within Maes's system is general to every module, as are the various parameters that effect the activation spread between modules. Converse to this, is the localised setting of ratings within AIMAE, as determined by each goal-strategy pairing individually. Furthermore, rather than a single measure effectively determining modular activation, AIMAE is able to make use of a number of different ratings for strategy selection and activation.

---

[33] Although there does exist the 'spread' of performance evaluations between strategies within a strategy frame, from the lowest level actions, up to the highest level goal.

I believe that the use of multiple specific parameters, i.e. localised rating sets, provides AIMAE with a more flexible and powerful means of controlling behaviour activation, than the use of activation energy by Maes, and below I outline how this occurs.

In accordance with requirement 2.5.iv, when a new goal is entered within the Control Plan that goal must be partially expanded, both in order for a time line to be created for it, and in order for possible opportunity taking to take place (see below). Such initial expansion, and indeed all goal expansion, tends to be carried out in a heuristic fashion within the Control Plan, via its use of ratings in a non-deterministic, stochastic manner[34]. Consequently, the choice concerning which rating types to use during strategy selection is of prime importance to the viability of an agent, especially where multiple rating sets exist (see Section 7.2.6). Such decisions concerning the criteria strategies are to be selected on are made within the Deliberation Module (see Appendix F Section 2.3.f).

Having then decided upon the required rating set and type(s), the Deliberation Module must then decide between the two main ways strategy selection can take place (see Appendix F Section 2.3.g), these being:

a. Deterministic Selection.

The most rudimentary method a system can employ in the selection of strategies is simply to carry out best first selection. Although this method guarantees that the best strategy will be chosen, a system using this approach may find that through aberrant behavioural results normally good strategies have their ratings lowered and are never used. Consequently, such an approach may give rise to behaviour which tends towards being stereotypical and predictable given a particular situation and goal, i.e. pure best first selection is deterministic.

b. Probabilistic Selection.

---

34 However, where such heuristic goal expansion is either not possible, or desirable, more reasoned decisions concerning strategy selection are possible within the Deliberative Module.

Conversely, strategy selection can be carried out in a more stochastic fashion, selection being probabilistically determined on the basis of a goal's associated strategies and their assigned ratings. For example, if a goal has three associated strategies with respective time ratings of 10, 20, and 30, then the probability ratio of each of those being selected is 3:2:1 (assuming that selection is aimed at finding the fastest strategy). Hence, unlike the above, this method does not give rise to behaviour which tends towards being stereotypical and predictable given a particular goal and situation. This is both good and bad depending on the goal in question and the current situation. In certain circumstances a system may need to be guaranteed that the best strategy is chosen, and in those circumstances this approach to strategy selection is unwise.

Unfortunately, this is not quite the end of the story. Because of the manner in which ratings propagate through a strategy frame, all but the 'lowest'[35] strategies will be rated in terms of other lower strategies, rather than directly initiated system behaviour. As such, it is conceivable that the 'higher' a strategy the more likely it is that its ratings may not be a wholly accurate portrayal of its performance. Secondly, consideration must be given to the manner in which run-time ratings are used to alter stored ratings, if not enough weight is given stored ratings then aberrant run-time ones may severely effect the system. Strategy selection should take account of both of these factors.

When a system employs deterministic selection, one way of taking into account the above two factors is to ensure that for a strategy to be chosen its ratings are greater than the ratings assigned to the goal's other strategies. The difference being a system set amount[36]. For example, if a goal has three associated strategies with assigned efficiency ratings of x, y, and z (respectively), and x is the highest, then for the first

---

[35] The terms 'low' and 'high' in relation to strategies within a strategy frame do not relate to their absolute Cartesian position within that frame, but rather how close they are to producing actual agent behaviour; what one might term their 'abstraction' level. Hence, a strategy could be utilised near the top level goal of a strategy but still be counted as 'low level' if it results in actual behaviour. Generally, strategies tend to maintain their level of abstraction regardless of the goals they are attached to, and the strategy frames they appear within.

[36] One might argue that this comparison would be extremely difficult, or arbitrary even, when carried out between quantitative and qualitative ratings. However, it must be remembered that this form of cross rating type comparison does not occur, comparisons are only carried out between the same rating type for each associated strategy!

strategy to be chosen x must be greater than y or z by the system set amount. If this is the case then the goal in question is expanded into sub-goals and actions via that first strategy, and this is done without the need for Deliberation Module intervention. However, if x is not greater than y or z by this system set amount, explicit deliberation must be used in deciding which strategy the Control Plan should utilise.

Converse to this is the use of a strategy's frequency rating to bias probabilistic selection. The more frequently a strategy has been used, the more accurately its ratings will reflect its performance, and hence the more probable should be its selection. Yet this does give rise to something of a dilemma. While aiming to select the best strategy, if selection is biased towards strategies with high frequency ratings then those with low frequency ratings may be overlooked even though they are actually better in terms of the chosen rating(s). In effect the use of frequency biasing makes the system more conservative.

Finally, it must be remembered that when more than one rating type is chosen for use in strategy selection, the relative importance those rating types are given must also be decided upon, regardless of the selection method used. Hence, in summary five main decisions are required (by the Deliberative Module - Appendix F Section 2) prior to performing actual strategy selection:

a. Deciding what rating set to use.

b. Deciding what ratings types from that set to use.

c. Deciding upon the relative importance's of those types.

d. Deciding upon the selection method.

e. Deciding whether to utilise any form of difference threshold, or frequency biasing?

f. Deciding how to combine rating types.

For obvious reasons, once a strategy has been chosen in service of a particular goal the pre-conditions for that strategy must be checked prior to any of its actions being executed. For instance, one pre-condition that must be met for all strategies is that the time available to the system for their execution is not smaller than the *time* rating assigned to them. Furthermore, in addition to such strategy pre-condition checking each action will also have pre-conditions which must be met, and therefore checked. However, because the real-world is continuously changing (depending of course of the level of abstraction that is used to view it), such checking is seen to occur not at the point of strategy selection, but when the system wishes to execute the particular action in question. In both cases when pre-conditions are found to be unmet the system must decide in what order unmet pre-conditions should be addressed and how, or whether to select another strategy. As mentioned earlier (see Section 7.2.4) such pre-condition checking and decision making are seen as being the preserve of the Deliberation Module (see Appendix f Section 7.2.3.e). Yet this being the case there may be times when the computational resources of the Deliberation Module are engaged in activities which preclude it from providing a correct assessment of pre-conditions. In such circumstances the Control Plan must go it alone (so to speak), either adopting strategies or attempting to execute actions, which may not be entirely congruent with the current state of the environment, the system's goals, and the system's dispositions. For instance, planning to go home by train only to realise later that the trains are on strike, or attempting to write with a pen without taking the lid off. Furthermore, such behavioural faux pas are in conjunction with the possible mistakes that may arise when conditions (both pre and extended) are not checked during action execution.

7.2.9 The Control Plan Revisited

In the previous eight sub-sections it has been shown how the AIMAE system intends to store plans, index those plans, rate them, and expand goals using them, however, it must be remembered that these tasks are to be achieved within a 'situated' agent, i.e. one which is goal directed, reactive, opportunistic, and temporally sensitive. The Goal Management Module utilises the control plan to accomplish all of these tasks along with the integration of ratings and run-time behaviour-based learning.

As previously mentioned, when a newly generated goal passes through the Attention Filter it is partially expanded so that goal adoption/rejection assessments may take place (Section 7.2.1) and a Termination Point calculated (Section 7.2.2). Both of these processes are carried out heuristically where possible and continue until the first executable action is encountered. Indeed, such heuristic expansion also occurs for older goals when a component specifies a sub-goal. For example, using the plan from Fig. 7.6, expansion would continue until action 1 was encountered. Since it is highly likely that the Control Plan will contain more than one goal at a time, it is also highly likely that it will have more than one action to chose from when deciding what to pass to the Execution Module. In such a situation the Control Plan must decide which action to select, and in service of which goal. This is done in the following manner:

a. For each possible action that may be chosen take the intensity value for every goal that specifies it, and add those values together to give a 'strength' rating for those actions. Use the results to order the list of actions.

b. Chose the action with the highest result from step (a). At this point the system now has an action which may potentially service more than one goal (depending of course on whether the action was specified by more than one goal).

c. Check the pre-conditions for the chosen action. If there exists a number of different pre-condition sets because of different goals specifying the action, check each set, and go on to step (d). If however, there exists only one set, either because only one goal specified the best action from (a), or because a number of goals specified exactly the same pre-condition set, skip step (d) and go on to step (e).

d. If there are a number of possibly choices as to which goal the action should address the system can do one of two things, which by experiment I hope to decide between:

i. Rate each instance of the action in terms of the intensity of its originating goal and the number of pre-conditions it has met. Chose the best action instance from the resultant list.

ii. Discard the failed action instances and pick from the remainder the one whose originating goal had the highest intensity.

e. Pass to the execution module the chosen action. Flag the goal(s) it is in service of, increasing their intensity values to help maintain coherence when the next action is chosen. Decrease the intensity of all other goals within the Control Plan, both for the sake of coherence and goal termination.

Once the above has been carried out, and any new goals have been assessed and posted it is repeated. At this point the Control Plan may attempt to send to the Effector Module (see Appendix F Section 3) an action that it cannot execute; either it interferes with what is already being done, or requires the output devices that are busy. Either way the Control Plan is notified of this and must decide whether to wait one cycle before re-submitting the chosen action, or chose a different action to execute. Such a choice should be made on the grounds of the 'strengths' of the various competing actions. If an action's 'strength' is above its nearest neighbour by a threshold amount the Control Plan will wait, otherwise it will choose the next action from the list in point (a) above (see Section 7.2.8). Similarly, when an action has been sent to the Effector Module, on the next cycle of the Control Plan if that action is still being executed, the system may either wait before choosing and then selecting another action, or attempt to execute one of the other pending actions. Such a decision being based on the intensity of the goal(s) with actions being executed. Hence, it is possible on each cycle of the Control Plan for it to send new actions to the Effector Module, even if the previously sent ones have not been completed. This is determined both by the actions that the Effector Module can carry out in parallel, and the intensities of extant system goals.

Finally, the Control Plan must also deal with goal terminations. A goal may terminate for two reasons, either it has been satisfied (as determined via condition checking), or its intensity has dropped to zero. While the former has been discussed, it has yet to be detailed how the latter occurs. If a goal specifies an action that cannot be met, or the goal is not highly intense, then that goal's action will not be selected for execution, and its intensity will be decreased by a small system set amount (see step e above). The longer such a state continues the lower the goal's intensity value will become. When this value drops to zero the system may either terminate the goal prematurely, or re-evaluate it. Re-evaluation is carried out first using the Control Plan's heuristic goal assessment mechanisms, and if not rejected at this stage, secondly by the Deliberation Module (see Appendix F Section 2.3.h). There is therefore no explicit mechanism for goal suspension. A goal can continue indefinitely within the Control Plan not being serviced, either because its intensity never drops due to Goal Generator re-activation, or because it gets re-evaluated.

It is hoped that such a Control Plan system will provided a basis for coherent behaviour in the face of changing environmental events and hence new goals, but also behaviour which may satisfy more than one goal at a time, i.e. opportunity taking. Furthermore, no explicit suspension mechanisms need to be postulated unlike with other similar systems, e.g. NML1, but at the same time goals which have become pointless are rejected. Finally, reinforcement learning is also supported in a fashion which is both intrinsic to the Control Plan mechanisms and determine by the agent's own behaviour.

7.3 Various Data Stores

Within the AIMAE system there exist a number of data stores, and these are as follows:

i. Perceptual Module Goal Store.

Stores goal after activation prior to passing through the attention filter or being deleted.

ii. Perceptual Module World Map.

Stores transitory information regarding the agent's domain environment, e.g. object x is at location y, as well as transitory statistical information. Within AIMAE such information is represented in a very basic STRIPs like form.

iii. Control Plan.

Holds information regarding the various states of extant system goals for use by Goal Management Module mechanisms.

iv. Deliberation Module Condition Store.

The extended conditions for all 'active' operators are held here, with those conditions belonging to the more intense goals being checked more than those belonging to less intense goals.

v. Domain Knowledge Store.

Stores information of a long term nature concerning the agent's environment, e.g. cars drive on the left hand side of the road. It is not clear how such information should be represented within AIMAE.

vi. Goal Lookup Table.

Stores information concerning what strategies may achieve what goals. It could be considered as a sub-part of the Domain Knowledge Store.

vii. Strategy Lookup Table.

Stores information concerning what operators achieve what strategies. It could be considered as a sub-part of the Domain Knowledge Store.

Each module should also be seen as having its own working memory.

7.4 Summary

The main focus of this paper has been to investigate the problems associated with the integration of situated activity and behaviour-based learning within a broad but shallow agent. The proposed AIMAE architecture provides a command and control system for just such an integration, utilising dynamically updateable ratings for the multi-dimensional ordering of strategies. Strategies that are used in the run-time expansion of goals interleaved with the execution of resultant actions. Not only will the proposed AIMAE system address the question as to whether ratings have a role in the integration of planning and learning, but it will also allow and exploration of the variety of ways in which such ratings may be utilised. For example, how run-time ratings are calculated, propagated, and merged with store ratings. Hopefully by examining the results of such variations within the same architecture a better understanding of how the effected mechanisms can be achieved, along with possible insights into equivalent human mechanisms.

Finally, the dynamically re-orderable scheduling mechanism allows not only for cyclical goal-expansion and re-assessment, but also for multiply active goals, where deliberative 'attention' may be focused on any of those goals depending on their relative 'intensities'. Consequently, it is conceivable that while the execution of one goal is being closely attended, that of another isn't. This may give rise to certain behavioural effects such as action slips. The following chapter outlines the variety of scenarios that may be used for the investigation of such a system, focusing on the scenario used by the Cognition and Affect Project, The Nursery.

8. POSSIBLE SCENARIOS

One of the biggest problems that beset earlier research in to command and control systems, was the limitation imposed on such work by the scarcity of computational resources. Consequently, we have seen how investigators tended to concentrate either on complex problem solving within simplistic environments, or simplistic problem solving within complex domains (Chapter 4). However, as the cost of computing power has decreased it has become increasingly viable for systems to exhibit complex reasoning within complex domains (Chapter 5). The problem is that because this dream is still not fully realisable the majority of systems have tended to focus resources in a deep, but narrow fashion, e.g. temporal reasoning at the expense of learning. While a step in the right direction, such systems are still therefore of limited use either for practical applications, or cognitive and psychological research. In Chapter 2 it was shown that the philosophy underpinning work carried out by the Cognition and Affect group takes a markedly different approach, attempting to integrate a wide variety of functions that an intelligent agent might require in a simple, but cohesive fashion. More specifically, the proposed AIMAE architecture aims to integrate goal directed planning and behaviour-based learning within a broad but shallow agent. For such an approach to result in a furtherment of scientific knowledge the environment in which resultant systems, e.g. AIMAE, are to be embedded must be of a real-world nature. Below a number of possible scenarios are put forward that satisfy this requirement.

8.1 The Nursery Scenario

The main scenario that the Cognition and Affect group have utilised, and the one that AIMAE will be implemented within, is the Nursery scenario (Sloman & Humphreys 1992; Sloman 1994). Simply, this comprises of a nurse having to look after a number of babies, babies which can get into all sorts of trouble if left to their own devices. The nursery itself comprises of equal size rooms set out to form a square. Thus, there may be 1, 4, 9, 16, etc. rooms in a nursery. Each room is separated from its neighbours by doors which may be open or closed, closed ones requiring keys to be opened, and may contain a number of objects such as food, medicine, or toys (see Fig. 8.1). Finally, all the babies and the nurse act in asynchrony with each other, and in real time. However, this version of the scenario does differ slightly from previous implementations used by the group:

i. There no longer exists a notion of ditches that babies can fall into.

ii. There is no notion of an independent camera that the nurse utilises to monitor areas of the nursery remote from its position; the nurse can only see objects within the same room as itself, but crucially it can hear babies in other rooms crying.

iii. The number of ways that babies can get into trouble has been extended such that the following may occur (some of which are new and some of which are common to all the nursery implementations):

  a. Babies can get hungry.

  b. Babies can fight, either because of room overcrowding, or for possession of a toy.

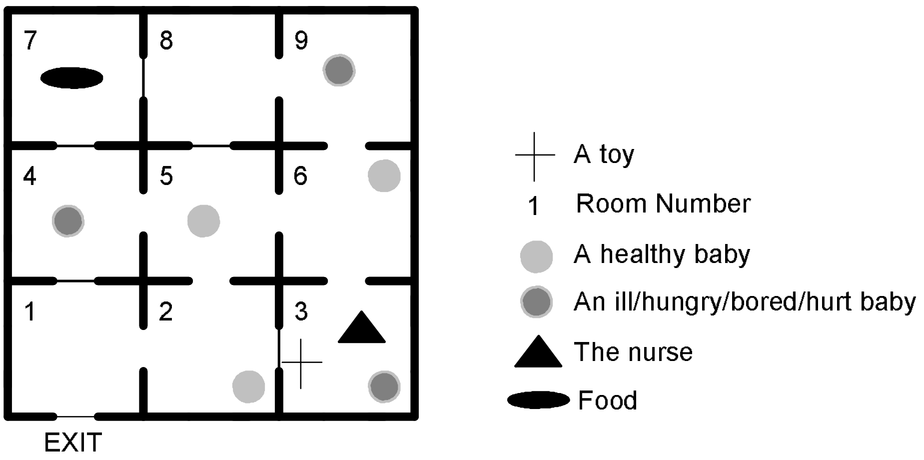
  c. Babies can get bored when they have nothing to play with.

  d. Babies can become ill.

  e. Babies can have accidents, such as falling over and getting hurt.

Let me give a simple example. In Fig. 8.1 it can be seen that there are six babies, one each in rooms 2, 3, 4, 5, 6, and 9. Three of those babies (those in rooms 3, 4, and 9) are in need of the nurse's assistance. Let us imagine that the baby in room 3 is bored, the baby in room 4 hungry, and the baby in room 9 ill. The nurse must decide not only the order that it should attend the babies but also how it will do so, taking in to account the intensity of each problem, and the potential ease of alleviating those problems. The trouble is that since the nurse is in room 3 it does not know what problems beset the babies in rooms 4 and 9, it only knows about the problems of the baby in its present room, room 3. Thus, the nurse must make a decision about what to do next on the basis of incomplete data. In such a situation it is likely that the nurse will deal with the bored baby in room 3 prior to the others. But is this wise? It is possible that the ill baby in room 9 is about to die and should thus be dealt with before the others, but how can the nurse know this? One method that the currently posited AIMAE system could utilise is to have associated with an unknown goal two strategies, one of which tells the system to investigate, and one of which tells it to carry out reasoning based on prior knowledge.

Fig. 8.1 - A Nine Room Nursery



Hence, the system should be capable of generating new goals through deliberation, and I see this as one of the crucial differences between agents with and without the equivalent of a Deliberative Module (see Appendix F Section 2). Those without have all of their top level goals generated directly via environmental events, whereas those with are capable of generating new top level goals in relation to other goals, rather than environmental events. Furthermore, in allowing specific information gathering and processing goals, strategies can also make use of them (e.g. Pryor 1991). For example, the nurse notices that a room is overcrowded and so realises that some of the babies must be moved to other rooms. However, having got this far an explicit information processing step must be specified within whatever strategy is being used so that the nurse may determine which babies to move into which rooms. This may itself require various information processing and gathering steps.

In summary, the nursery scenario provides an environment which is not only complex, dynamic, and unpredictable, but also one that requires that the nurse rank and swap between goals as new problems and opportunities arise, i.e. it shares the same essential characteristics as the real world. In short, it is a simple, understandable and controllable real-world environment, allowing investigation into the use of ratings for the integration of planning and learning, as well as providing possible insights into why unwanted maladaptive behaviour may occur within a system such as AIMAE.

8.2 Other Possible Scenarios

Aside from the nursery scenario in which the AIMAE system is to be embedded there are many other possible scenarios, all of which provide real-world environments, of potential commercial, civilian, and military uses. For example:

i. Automated air traffic control systems.

ii. Automated battlefield command and control systems.

iii. Logistics planning.

iv. Automated medical care systems, e.g. intensive care (as used by Hayes-Roth).

v. Automated guided tours systems.

vi. Complex factory control systems, e.g. for oil refineries, or power stations.

vii. Stocks and shares management.

Any environment that requires a system capable of dealing with multiple conflicting goals, in a resource limited fashion, and learn from past performances, is a prime example of the sort of environment that AIMAE is designed to be embedded within.

9. CONCLUSION

9.1 Real-World Command & Control

The main focus of this paper has been to examine techniques by which a real-world embedded agent is able to produce goal directed behaviour, sensitive not only to environmental events, but also past experiences. Such an agent will be subject to a variety of constraints imposed by both the environment, as well as its own physical and cognitive capabilities, where these constraints should be seen as determining to a large extent the requirements that the design for such an agent must realise. In the course of investigating these constraints and the requirements for embedded computation that they give rise to, a number of techniques and systems have been explored as summarised below.

i. Classical planning approaches (Chapter 4).

ii. Reactive planning approaches (Chapter 4).

iii. Model-based reasoning systems (Chapter 5).

iv. BDI systems (Chapter 5).

v. Dynamic control reasoning (Chapter 5).

vi. Nouvelle AI approaches (Chapter 6).

However, these approaches and the systems that have been used to implement them are not without their faults. First and foremost, the representational approach used in the transcription of perception into action determines a plethora of philosophical and practical problems (see Sections 4.1 & 6.2). Most notably, the use of abstract symbolic representations in conjunction with classical planning techniques has proven to be unviably expensive in terms of computational resources (Section 4.2), even with the many techniques that have been developed to reduce these costs (Appendix B). However, the use of purely reactive systems (Section 4.4) is also fraught with problems. The most obvious of these is that the simplistic methods used to generate action cannot account for behaviour which extends in time, such as driving, nor take advantage of information which is removed either temporally, or physically. Given this, it is obvious that for an agent to remain viable within a real-world environment it must combine the advantages of both of these approaches, balancing deliberation against reaction, and producing goal directed behaviour which is in accordance with the environment and not just its pre-stored plans. Systems which attempt to achieve this type of balance, while utilising abstract symbolic representations, are termed Integrated Architectures (Chapter 5).

Integrated approaches to situated activity that provide goal directed, but reactive behaviour, have tended to fall into three categories, those that use using model-based reasoning techniques, those that use the BDI formalism, and those that carryout a degree of dynamic control reasoning. Model-based reasoning techniques (Section 5.1) have the potential not only for producing biologically plausible situated activity that can alter with respect to learning, but also for supporting meta-management and emotional processes. In this respect they provide one method for integrating planning and learning within a real-world embedded agent. However, such techniques do not address implementation problems, such as how goals should be generated and managed, nor how agent knowledge should be managed.

An alternative approach which does address some of these issues is that of the BDI formalism (Section 5.2), as implemented within PRS (Appendix D Section 2) and NML1 (Section 5.3). Using this an embedded system is viewed as a rational agent with beliefs, desires and intentions, and various mechanisms are suggested for manipulating these in the production of behaviour. While both PRS and its extension NML1 address issues such as goal generation and goal management, they fail to account well for learning, time management and memory management. With respect to time management (and possibly learning), BB1 (Section 5.4) and GUARDIAN (Appendix D Section 3) extend NML1s ability for dynamic control reasoning via the use of a control plan and associated management mechanisms. Yet, this still leaves problems in terms of integrating learning into behaviour production, and efficiently managing memory resources.

A complete change in the approach to producing viable goal directed activity is offered within the Nouvelle AI paradigm (Chapter 6). Using this, either the traditional manner of representing the world is done away with (Section 6.1), e.g. PENGI (Appendix E Section 1), or the traditional manner of decomposing an agent's internal architecture is done away with (Section 6.3), e.g. Situated Automata (Appendix E Section 3), or both are, e.g. Subsumption Architectures (Appendix E Section 2). However, the removal of explicit abstract symbolic representations leads to many of the problems that hound more traditional reactive architectures (Section 6.2), e.g. goal management, behaviour which persists in time, and learning. Similarly, a behavioural decomposition of an agent leads to problems (Sections 6.2 & 6.4 respectively) such as information fusion, processing redundancy, and module co-ordination. This last problem is eloquently addressed within Maes's Spreading Activation Networks (Appendix E Section 4), although these, like the other Nouvelle AI systems, fail to provide sufficient goal generation and management mechanisms.

AIMAE is effectively an extended amalgamation of both NML1 and GUARDIAN. As such, it benefits from the same advantages these do in terms of goal management, but crucially also provides support for learning in the form of ratings and their manipulating mechanisms. In utilising ratings for the selection of plan components AIMAE can be seen as equating well with Maes's approach to situated activity, however it advances this approach by making selection much more flexible. Thus, the use of ratings in conjunction with a Brooksian approach to the Execution Module, means that AIMAE should be seen as integrating not only reaction, deliberation and learning, but also traditional and Nouvelle AI approaches to embedded computation. Finally, AIMAE should be seen as satisfying all requirements as laid out within Chapter 3 (bar requirement 3.6.viii) and providing an architecture within which much investigation and design space exploration may take place.

9.2 The AIMAE System

The proposed AIMAE architecture provides a command and control system that supports the production of goal directed behaviour within a real-world embedded 'broad but shallow' agent. In keeping with this, it provides to differing degrees of 'depth', mechanisms for many of the functions that we might ascribe to humans and other 'higher' lifeforms. For example, perception, goal scheduling, goal planning, learning, reasoning and action execution. However, being primarily concerned with the integration of planning and learning, the design for AIMAE has focused most closely on the handling of goals within the Goal Management Module. This module plays a central role within AIMAE, carrying out the following three tasks in a cyclical fashion on surfaced goals:

i. Goal Assessment.

ii. Goal Expansion.

iii. Goal Scheduling.

Further to its role as an integrated planner and scheduler, the Goal Management Module performs a function which is central to the theme of this paper, that is learning. In conjunction with Goal Management mechanisms AIMAE utilises two cross-indexing lookup tables, the Goal Lookup Table and Strategy Lookup Table, to heuristically expand Control Plan goals into partial plans. Of crucial importance in this process is the use of both quantitative and qualitative ratings to index strategies within the Goal Lookup Table (Section 7.2.6). These ratings aid the system in deciding between the various strategies associated with a goal by ordering those strategies in a multi-dimensional, context dependent manner. Although this use of strategies and ratings is superficially very similar to the manner in which Maes has implemented Spreading Activation Networks there are a number of crucial differences:

i. Strategy selection is not based upon a threshold being reached, but rather is probabilistically biased (or deterministically aimed) towards the strategy with the best ratings.

ii. There is no notion of a global threshold determining strategy selection. Instead, selection is always carried out with respect to local values, i.e.. the ratings associated with the other strategies assigned to the goal in question.

iii. There is no notion of spreading activation.

iv. In spreading activation networks activation spread is controlled via a number of global parameters, but in AIMAE ratings are always local to the particular goal-strategy pairing in question.

Hence, the use of ratings within AIMAE offers a clear advantage over the more restrictive use of activation levels within Maes's spreading activation networks. Furthermore, because strategies are assigned to goals and selection occurs in accordance with goals, AIMAE is not required to search its whole 'database' to identify possible applicable strategies. This is unlike the selection of 'knowledge areas' within PRS, RAPs, NML1, and Guardian, and thus AIMAE also offers clear advantages over the memory management and information retrieval mechanisms utilised within other approaches such as Universal Planning, PRS, RAPs, or NML1.

In addition to this role as a technique for indexing stored information, ratings provide AIMAE with a means of integrating a degree of learning into planning (Section 7.2.7). As strategies are executed in the realisation of goals they are characterised along a number of dimensions which are then employed to alter the stored context dependent ratings for particular goal-strategy pairings (see Appendix F Section 5 for an example). Since stored ratings play a determining role in the production of agent behaviour, it can be seen that altering those ratings on the basis of run-time execution provides AIMAE with a means of achieving both reinforcement and context learning in a manner which is integral to system planning. Furthermore, unlike many learning systems where an increase in agent knowledge can lead to a decrease in agent response times (Minton 1985), e.g. SOAR and other case-based planners (Appendix B Section 7 & 8), an increase in agent knowledge via the alteration of stored ratings is unlikely to lead to such decreases, but is likely to lead to more viable behaviour.

In summary therefore, AIMAE provides mechanisms for the asynchronous and parallel generation of goals on the basis of system concerns, extending the goal generation mechanisms provided by most systems, such as PRS, RAPs and spreading activation networks. More importantly, however, AIMAE extends the management of goals within such systems, as well as NML1, in a number of ways:

i. AIMAE supports a notion of time and scheduling is partially based upon this and the approach of deadlines.

ii. AIMAE is able to pursue a number of goals concurrently without any need to explicitly suspend and re-activate those goals. This support for multiply active goals gives rise to the next point.

iii. AIMAE supports three types of opportunity taking:

iv. Opportunity taking in the sense of noticing when a goal has been spontaneously achieved by the environment, or conversely, made unachievable.

v. Opportunity taking concerning the partial achievement of more than one goal via the execution of one action.

vi. Opportunity taking with respect to taking advantage of environmental, or system, changes to pursue the achievement of an adopted goal.

vii. AIMAE provides support for both performance related learning and contextual learning.

Finally, by integrating a wide variety of techniques for embedded computation (from both traditional and nouvelle AI), AIMAE should be seen as satisfying all requirements as laid out within Chapter 3 (bar requirement 3.6.viii) and providing an architecture which achieves the above in a manner which cannot simply be characterised as a restrictive following of any one design approach. While it might be argued that a number of the features AIMAE attempts to provide are achieved via other systems, for example learning by SOAR and time management by GUARDIAN, none of the examined systems offer all of these features within a unifying framework. AIMAE does, and provides an architecture which within which much design space exploration may take place. As such AIMAE should be seen as a major advance over other real-world embedded systems.

9.3 Insights Into Psychology

However alluring, there is something disturbingly scary with the realisation that as science progresses machines will grow to replace humans in a plethora of roles; a prospect already very much in evidence with, for example, factory production lines, and aircraft flight control systems. The images films like West World and Terminator portray are all the more profound not because of their depictment of machines with human-like intelligence, but because of the demonstration of how such intelligence may be turn out to be the bane of mankind, usurping humans from the position of 'top-dog' to one of machine subordination. Indeed, in many respects we are already slaves to the machine, a process started with the industrial revolution and still far from over. However, just as a sword may be double-edged so too is the rise in AI technology, giving mankind the option of creating a whole new investigative tool into the psychological workings of intelligent animals such as ourselves. So far I have attempted to develop ideas concerning the integration of planning and learning. However, the approach utilised has been with the ultimate aim of achieving a design from which insights into the cognitive behaviour of agents such as ourselves may be gained. As A.C. Clark & G. Lee (Cradle - Futura Publications, 1989, p.131) write...

> "But it's virtually impossible for human beings, even with a common language and developed communications skills, to truly describe their feelings. How could we even know or appreciate, for example, a dolphin's sense of loneliness? In our maudlin way we ascribe to them human emotions, which is ridiculous.' He paused for a moment to think. 'No,' he continued, 'it's more fruitful to conduct scientific inquiry at levels where we can understand the answers. In the long run, I believe that knowing how these creatures function, in the scientific sense, is more likely to lead us to their emotional quotients than conducting psychological experiments whose outcome cannot be interpreted."

It is hoped that with AIMAE we stand in a position to ask questions of our design such as what are emotions, how do they interact within the system, and where do they come from? Up until recently, the study of how goals and emotions interacted was a poorly `attended' area. However, notable theories include work by Simon (1967), Sloman & Humphreys (1992), Johnson-Laird (1989) and others. It seems that a vast majority of the theories on motivation and emotion fall into two main categories. Those that believe that emotions are an emergent product of the complex cognitive architecture that supports mechanisms of arbitration between conflicting goals in a limited capacity processing system, and those that believe that emotions have specific in-built, i.e. innate, mechanisms:

i. Emergent:

Simon (1967), believed that an interrupt mechanism that allowed a non-active goal to become active, i.e. interrupting the currently active goal, was a necessary feature that stopped a serial architecture of goal `management' from becoming single minded in nature. He points towards the fact that there is a close relationship between this interrupt mechanism and emotional behaviour, whereby the interrupt mechanism when activated causes an aroused emotion.

This idea that emotions arise due to mechanisms whose primary purpose is non-emotional is further supported by Sloman & Humphreys (1992), who see emotions as being an emergent product of the architecture that must underlie the mechanisms for motivation and attention within a processor limited agent. However, not wanting to miss-represent, Sloman & Humphreys (1992, p.2) do state that "...this is not to deny that there may be some specific emotions... that are produced by specific mechanisms".

ii. Non-Emergent:

Both Johnson-Laird (1989), Izard (1993) and to some extent Kuhl (1992), advocate that emotions are a product of specific emotional mechanisms within the brain. This may well be the case, in fact it is hard to imagine that since emotions do play a role in behavioural mechanisms that their existence is a mere `oddity' of the system rather than one which has been designed into it. Neurophysiological evidence also seems to point towards the fact that there exist within the brain, sites designed specifically to deal with emotions (e.g. Amygdala and other Limbic System structures). However, what these studies do not divulge is whether the emotions that these sites seem to operate on are emergent properties from the system or properties which emerge from specific brain areas. Just because there exist mechanisms for dealing with emotions doesn't mean that the emotions actually arise specifically from mechanisms in the first place. Emotions may arise as a by-product, or be specifically produced, in either case there will need to be mechanisms/modules for dealing with them (cf. thrashing within computer systems).

Within the preceding chapters an in-depth investigation of the plethora of constraints that impinge upon real-world embedded agents was followed by an exposition of the myriad of command and control systems that have arisen to combat those constraints. However, none of those techniques and systems have as yet allowed science to turn fiction into fact, and empower computers with minds. Although, the posited AIMAE system is not the ultimate culmination of computer intelligence, it does however specify the design for an autonomous agent's command and control system that allows the elicitation of a number of hypothetical behaviours, both adaptive and mal-adaptive. It is thus one more step in machine-kinds long walk in to consciousness.

10. REFERENCES

11. APPENDICES A - E